Eötvös Loránd Tudományegyetem
Természettudományi Kar

A sztochasztikus gradiens módszer

BSc Szakdolgozat

Írta:
Mészáros Joshua
Matematika BSc
Alkalmazott matematikus szakirány

Témavezető:
Dr. Izsák Ferenc
Egyetemi Docens
Alkalmazott Analízis és
Számításmatematikai Tanszék

Budapest, 2025

Köszönet nyilvánítás

Szeretnék köszönetet mondani mindenkinek, akitől segítséget kaptam ennek a dolgozatnak a megírásához, különösen témavezetőmnek Dr. Izsák Ferencnek.

Tartalomjegyzék

1. Bevezetés

Az utóbbi évtizedben, a neurális hálózatok hatalmas fejlődésen mentek keresztül, népszerűségük egyre inkább nő, már a mindennapi életben is használjuk a mesterséges intelligenciát. A ChatGPT, az önvezető autók, a kép és hang generáló intelligenciák, a rendszámfelismerő kamerák mind, ezt a jelenleg is fejlődésben lévő technológiát használják.

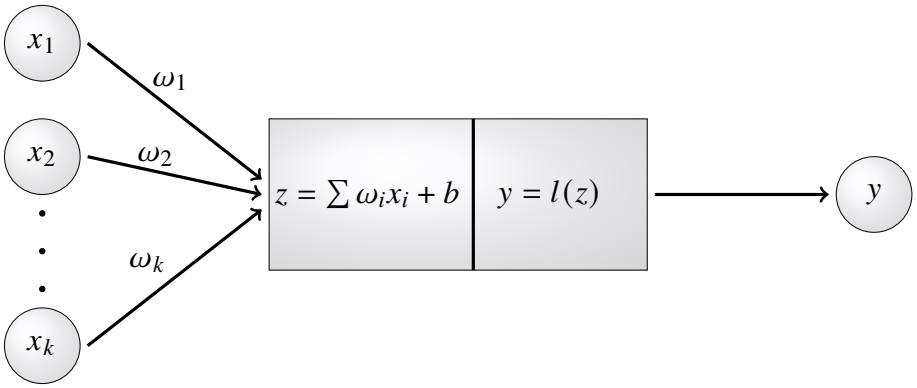
Szakdolgozatomban három fő komponens jelenik meg. A gyakorlati komponensben bemutatom a mesterséges neurális hálózatok alapjait, először az elemi neuront, majd a háló szerkezetét. Felvázolom, hogy ezeket matematikailag milyen függvényekkel reprezentáljuk, továbbá bemutatom a gyakorlatban használt backpropagation algoritmust a gradienst kiszámolására. Az elméleti részben először felvázolom a gradiens fogalmát, a minimális irány egyértelműségét (3.1 Bizonyítás saját eredmény), majd a gradiens módszert, és annak tulajdonságait. Itt a fix lépéshosszúságú módszer által adódó görbe konvergenciáját vizsgálom, ahol a 4.2 és 4.3 Bizonyítások is saját eredmények. Mindezek alapján bemutatom a sztochasztikus gradiens módszert, és annak tulajdonságait. Megvizsgálom az iteráció konvergenciáját erősen konvex majd általános függvények esetén, ezeken belül kitérek az konstans és változó lépéshosszúságú módszerekre. A dolgozat utolsó fejezetében, mutatok pár egyszerű szimulációs példát a korábbiakban leírtak bemutatására. Az egyes komponenseken belül, törekedtem arra, hogy egy fix formalizmust alkalmazzak a könnyebb átláthatóság kedvéért.

A dolgozatomhoz meghatározó részben Léon Bottou, Frank E. Curtis, Jorge Nocedal - Optimization Methods for Large-Scale Machine Learning [1] irodalmat használtam, többek között a neurális hálózatokat leíró függvények bemutatásához, a gradiens és a sztochasztikus gradiens módszerek bevezetéséhez, illetve a sztochasztikus gradiens módszer analíziséhez használt feltételek, tételek, állítások és lemmák is innen származnak. Mindezeken túl, több internetes forrást is igénybe vettem a témakör megértéséhez, illetve a szimulációk értelmezéséhez. Az optimalizációs eljárásokról általánosan és a dolgozatban fix lépéshosszúságú gradiens módszerként definiált Cauchy-módszerről [2] oldalon olvastam. A neurális hálózatok tanítási gyakorlatáról [3] oldalról tájékozódtam, továbbá a backpropagation algoritmust a [5] videóból értettem meg. A szimulációk elkészítéséhez [4] előadás tartalmából indultam ki, a konkrét szimulációs kódok egy részének generálására, pedig a ChatGPT mesterséges intelligenciát használtam.

2. Neurális hálózatok

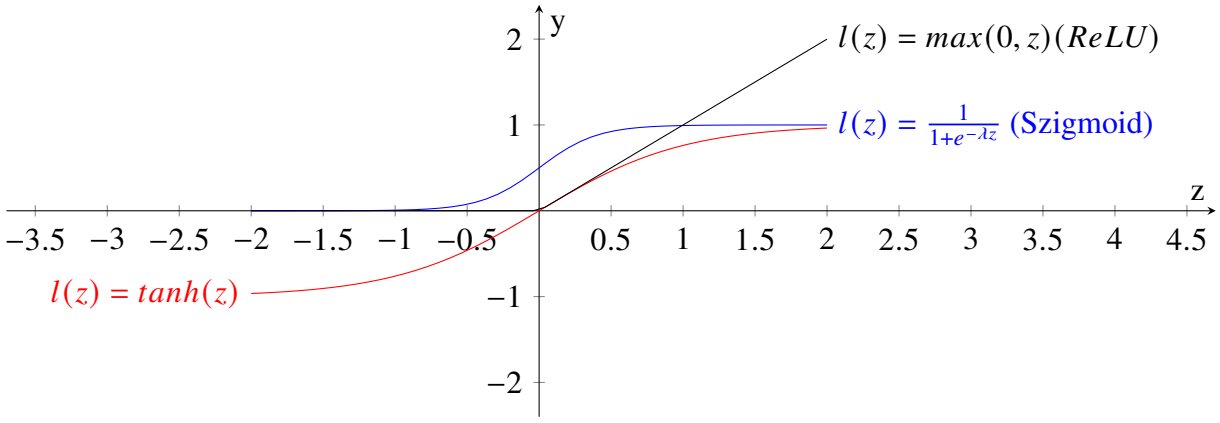
2.1. Elemi neuron

A gradiens, illetve sztochasztikus gradiens módszer egyik fő alkalmazási területe a neurális hálózatatok tanítása, optimalizálása. Egy neurális hálózat egy $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ függvény, melynek input értéke egy n-dimenziós vektor, mely a feldolgozandó adatot tartalmazza, output értéke egy m-dimenziós vektor, mely az általunk elvárt eredményt hordozza. A neurális hálózat tanításánál az a feladatunk, hogy beállítsuk a háló belső paramétereit úgy, hogy az általunk definiált feladatot hajtsa végre. A hálózat elemi neuronokból épül fel melyeknek van k-db bemenete, ezekhez hozzá vannak rendelve ω_i súlyok. Jelölje a súlyvektort (sorvektor) ω, a bemeneti vektort pedig jelölje x (oszlopvektor). A neuron először kiszámolja $\omega \cdot x + b$ értéket, ahol b egy konstans eltolás, majd átadja az aktivációs függvénynek: $l : \mathbb{R} \longrightarrow \mathbb{R}$. Ennek az értéke lesz a neuron kimeneti értéke. A b konstanst álatában úgy reprezentálják, hogy a bemeneti x vektor dimenzióját növelik 1-el, és ennek konstans 1 értéket adnak, így $b = \omega_{k+1}$. Egy elemi neuront mutat az 1. ábra.



1. ábra. Elemi neuron k elemű bemenettel, $\omega_1, ..., \omega_k$ súlyokkal, b konstans eltolással, és $l(z)$ aktivációs függvénnyel.

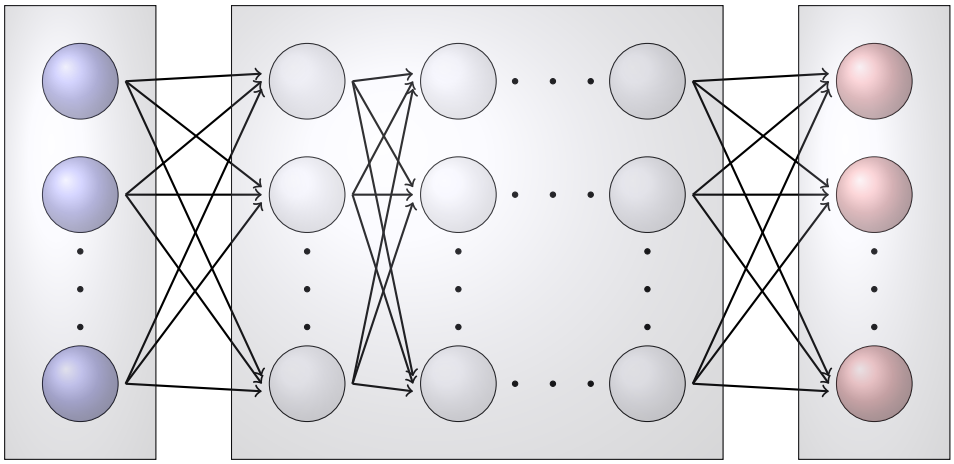
Az aktivációs függvényt a háló feladatának függvényében választják meg. A leggyakrabban használt típusok az alábbiak (2. ábra):



2. ábra. Szigmoid, ReLU és tangens hiperbolikusz aktivációs függvények.

2.2. A neurális hálózat felépítése

A neurális hálózat rétegekbe rendezett neuronokból épül fel. Egy réteg minden neuronjának kimenete hozzá van kapcsolva a következő réteg minden neuronjának bemenetére. Az össze nem kötött neuronokat 0 súlyú éllel reprezentáljuk. A 3. ábrán egy neurális hálózatot láthatunk.



Bemeneti réteg Rejtett réteg Kimeneti réteg

3. ábra. Neurális hálózatok felépítése.

2.3. A neurális hálózatokat leíró függvények

Jelölje $\mathcal{W}_i : \mathbb{R}^{d(i-1)+1} \longrightarrow \mathbb{R}^{d(i)+1}$ az (i)-edik réteghez tartozó lineáris leképezést, ahol $d(i)$ az i-edik réteg neuronjainak száma, azaz $\mathcal{W}_i$ egy $\mathbb{R}^{d(i)+1 \times d(i-1)+1}$ mátrix, melynek j-edik sora az i-edik réteg, j-edik neuronjába bemenő élek súlyait tartalmazza, beleértve a b_j konstans eltolást.

$$\mathcal{W}_i = \begin{bmatrix} \omega_{1,1} & \omega_{1,2} & . & . & . & \omega_{1,d(i-1)} & \omega_{b_1} \\ \omega_{2,1} & \omega_{1,2} & . & . & . & \omega_{2,d(i-1)} & \omega_{b_2} \\ . & & . & & & . & . \\ . & & & . & & . & . \\ . & & & & . & . & . \\ \omega_{d(i),1} & \omega_{1,2} & . & . & . & \omega_{d(i),d(i-1)} & \omega_{b_{d(i)}} \\ 0 & 0 & . & . & . & 0 & 1 \end{bmatrix}$$

Legyen $y^{(i)} \in \mathbb{R}^{d(i)+1}$az i-edik réteg által generált vektort, illetve $y_j^{(i)}$ ennek a j-edik neuronjához tartozó koordinátáját. Jelölje $L_i : \mathbb{R}^{d(i)+1} \longrightarrow \mathbb{R}^{d(i)+1}$ az (i)-edik réteg aktivációs függvényét, továbbá $l_j : \mathbb{R} \longrightarrow \mathbb{R}$ jelölje ennek j-edik koordinátafüggvényét. Ekkor:

$$L_i = \begin{bmatrix} l_1 & l_2 & . & . & . & l_{d(i)} & 1 \end{bmatrix}^T$$

Ezek alapján felírhatjuk az $(i-1)$. és az (i). réteg kimenetei közötti összefüggést:

$$y^{(i)} = L_i(\mathcal{W}_i y^{(i-1)})$$

Legyen a neurális hálózatunk r rétegű, illetve legyen $y^{(0)} = x$. Ekkor a $f(x)$ az alábbi rekurzióval írható fel:

$$f(x) = y = y^{(r)}$$

ahol

$$y^{(i)} = L_i(\mathcal{W}_i y^{(i-1)}) : i = 1, ..., r$$

A neurális hálózat tanításánál keressük azokat a súlyparamétereket, melyek az általunk elvárt működést biztosítják. Ehhez adottak olyan bemeneti vektrorok melyek esetén tudjuk az elvárt kimenet értéket. Tehát adott egy S halmaz, melynek elemei (x_i, y_i') vektorpárok. Ezt a halmazt három részre oszhatjuk, training set, validation set és test data set. A training set közvetlenül a háló tanítására szolgál. Ezeket a vektorokat használjuk a minimum kereséséhez a gradiens, illetve sztochasztikus gradiens eljárásokban. A validation set arra szolgál, hogy ellenőrizzük a tanítás során, hogy miképp viselkedik a háló olyan adatok esetén, amelyet nem használunk a gradiens módszerben. Ezzel lehet elkerülni az úgynevezett overfittinget, amely esetén a training set elemeire nagyon jó választ ad a háló, ellenben más vektorokra az elvártnál rosszabbat. Az a célunk, hogy a háló általálnosítson (generalization) és ne memorizáljon (memorization). A test data set pedig a tanítás után használt teszthalmaz, tehát olyan vektorok, amelyeket semmilyen értelemben nem használtunk a tanítási eljárás során.
Legyen e a hálóban lévő élek és neuronok számának összege és legyen $n = d(0) + 1$ azaz x_i input vektorok dimenziója. $\Omega \in \mathbb{R}^e$ jelölje az élek és eltolások paraméterezését, továbbá legyen $h(\Omega, x) : \mathbb{R}^{e+n} \longrightarrow \mathbb{R}^m$, ahol $m = d(r) + 1$, azaz az output vektor dimenziója. $h(\Omega, x)$ az $f(x)$ függvény értékét adja az f Ω paraméterezése esetén. Célunk, hogy a háló által adott eredmény és az elvárt eredmény közötti eltérés minimális legyen. Ennek a mérésére például alkalmas a négyzetes hiba. Tehát keressük:

$$min\left\{\frac{1}{|S|}\sum_{(x_i,y_i')\in\mathbb{R}} \left\|h(\Omega, x_i) - y_i'\right\|^2 : \Omega \in \mathbb{R}^e\right\}$$

minimális várható hibát. Erre ad egy megoldást a gradiens, illetve a sztochasztikus gradiens módszer, melyet a következő fejezetekben mutatok be.

3. A gradiens

3.1. Minimális irány

3.1. Definíció. Legyen $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ differenciálható függvény, továbbá $v \in \mathbb{R}^n$ az alábbi határértéket:

$$\lim_{\delta \longrightarrow 0} \frac{g(t + \delta v) - g(t)}{\delta}$$

a g függvény t pont beli v szerinti iránymenti deriváltjának nevezzük. Ezt az értéket megkaphatjuk $\nabla g(t) \cdot v$ alakban, ahol ∇g a g függvény gradiense.

3.2. Definíció. Legyen $g : R^n \longrightarrow R$ differenciálható függvény továbbá
$v_m = argmin\{\nabla g(t) \cdot v : \|v\| = 1, v \in R^n\}$. Ekkor a v_m vektort a g függvény t pont beli minimális irányának nevezzük.

3.1. Állítás. Legyen $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ differenciálható függvény és $\nabla g(t) \neq 0$. Ekkor v_m egyértelmű és értéke: $-\frac{\nabla g(t)}{\|\nabla g(t)\|_2}$.

3.1. Bizonyítás. A Cauchy–Bunyakovsky–Schwarz (CBS) egyenlőtlenség miatt tudjuk, hogy: $|x \cdot y| \leq \|x\|_2 \|y\|_2$ tehát a lehetséges legkisebb érték, amit az egyenlőtlenség megenged az: $x \cdot y = -\|x\|_2 \|y\|_2$, tehát $v_m = -\frac{\nabla g(t)}{\|\nabla g(t)\|_2}$ esetén :

$$-\|\nabla g(t)\|_2 \cdot 1 = -\|\nabla g(t)\|_2 \cdot \|v_m\|_2 \leq \nabla g(t) \cdot v_m = \nabla g(t) \cdot -\frac{\nabla g(t)}{\|\nabla g(t)\|_2} = -\frac{\|\nabla g(t)\|_2^2}{\|\nabla g(t)\|_2} = -\|\nabla g(t)\|_2,$$

azaz $v_m = -\frac{\nabla g(t)}{\|\nabla g(t)\|_2}$ valóban minimális irány. A következőkben belátjuk, hogy ez egyértelmű. Tegyük fel, hogy $\exists v_m, v'_m$, hogy $v_m \neq v'_m$ és mindkettő egyaránt minimális irány, és $v_m = -\frac{\nabla g(t)}{\|\nabla g(t)\|_2}$. Ekkor az előzőek alapján tudjuk, hogy:

$$v_m \cdot \nabla g(t) = -\|\nabla g(t)\|_2$$

és

$$v'_m \cdot \nabla g(t) = -\|\nabla g(t)\|_2$$

továbbá legyen $v_\Delta = v_m - v'_m$:

$$v_\Delta \cdot \nabla g(t) = (v_m - v'_m) \cdot \nabla g(t) = v_m \cdot \nabla g(t) - v'_m \cdot \nabla g(t) = -\|\nabla g(t)\|_2 - (-\|\nabla g(t)\|_2) = 0$$

azaz $v_\Delta \perp \nabla g(t)$ ezért $v_\Delta \perp v_m$-re is, tehát felhasználva a merőlegességet:

$$1 = \|v'_m\|_2^2 = \|v_m + v_\Delta\|_2^2 = \|v_m\|_2^2 + \|v_\Delta\|_2^2 = 1 + \|v_\Delta\|_2^2$$

amiből következik hogy: $\|v_\Delta\|_2 = 0 \Rightarrow v_\Delta = 0 \Rightarrow v_m = v'_m$, ami ellentmondás. □

3.2. Gradiens számolása backpropagation módszerrel

Szeretnénk kiszámolni a hibafüggvény gradiensét, azaz adott az első fejezetben tárgyalt $h(\Omega, x) : \mathbb{R}^{e+n} \longrightarrow \mathbb{R}^m$ függvény, és keressük $C(h(\Omega, x), y') : \mathbb{R}^{e+n} \longrightarrow \mathbb{R}$ gradiensét, ahol C a hibafüggvény. Ebben az esetben x vektort paraméternek tekintjük és a függvény változói, ami szerint keressük a gradienst azok az $\omega \in \Omega$ súly értékek. Jelölje $\omega_{u,v}^{(k)}$ az $\mathcal{W}_k$ leképezés (u, v) koordinátájának értékét. Korábban láttuk, hogy:

$$h(\Omega, x) = y = y^{(r)}$$

ahol

$$y^{(0)} = x$$
$$y^{(i)} = L_i(\mathcal{W}_i y^{(i-1)}) : i = 1, ..., r$$

Tekintsük először $y_j^{(i)}$ értéket, ami L_i j-edik koordinátafüggvényének kimeneti értéke. Ezt szeretnénk deriválni $\omega_{u,v}^{(k)}$ szerint. Tudjuk, hogy:

$$y_j^{(i)} = l_j^{(i)}(z_j^{(i)}) = l_j^{(i)}(\omega_j^{(i)} y^{(i-1)}) = l_j^{(i)}\Big(\sum_{m=1}^{d(i-1)+1} \omega_{j,m}^{(i)} y_m^{(i-1)}\Big)$$

Tehát a láncszabály alapján, ahol $k < i$:

$$\frac{\partial}{\partial \omega_{u,v}^{(k)}} y_j^{(i)} = \frac{\partial}{\partial \omega_{u,v}^{(k)}} l_j^{(i)} = \frac{\partial}{\partial z_j^{(i)}} l_j^{(i)} \cdot \sum_{m=1}^{d(i-1)+1} \omega_{j,m}^{(i)} \frac{\partial}{\partial \omega_{u,v}^{(k)}} l_m^{(i-1)}$$

rekurzió írható fel. Észrevehetjük, hogy csak a szummában lévő tagok értékei függnek $\omega_{u,v}^{(k)}$-tól, ezért, ha $k = i - 1$, akkor:

$$\frac{\partial}{\partial \omega_{m,v}^{(i-1)}} l_m^{(i-1)} = y_v^{(i-1)}$$

azaz megegyezik az előző réteg v-edik neuronjának aktivitásával. Ez azért nagyon előnyös, mert a parciális deriváltak számolásánál minden rekurzióban az azt megelőző réteg szerinti deriváltak rögtön számolhatóak.

A backpropagation lépései:

1. Kiszámoljuk $C(h(\Omega, x), y')$ -et, ezzel megkapjuk a $y^{(i)}$, $z^{(i)}$ vektorokat minden $i \in \{1, 2, ..., r\}$.
2. Meghatározzuk $C(y^{(r)}, y')$ parciális deriváltjait minden $y_j^{(r)}$ $j \in \{1, 2..., d(r)\}$ szerint.
3. Majd a fenti rekurzió segítségével meghatározunk minden $\omega_{u,v}^{(k)}$ szerinti parciális deriváltat.

Tekintsük az alábbi példát, amin bemutatom a backpropagation lépéseit. A bemeneti és kimenti vektoraink legyenek 2-2 dimenziósak és egyszerűség kedvéért a b (bias) értékek legyenek fixen 0 értékűek elkerülve így a vektorok konstans 1-el való bővítését. Az aktivációs függvény legyen szigmoid ($\lambda = 1$):

$$l_j^{(i)}(z_j^{(i)}) = \frac{1}{1 + e^{-z_j^{(i)}}}$$

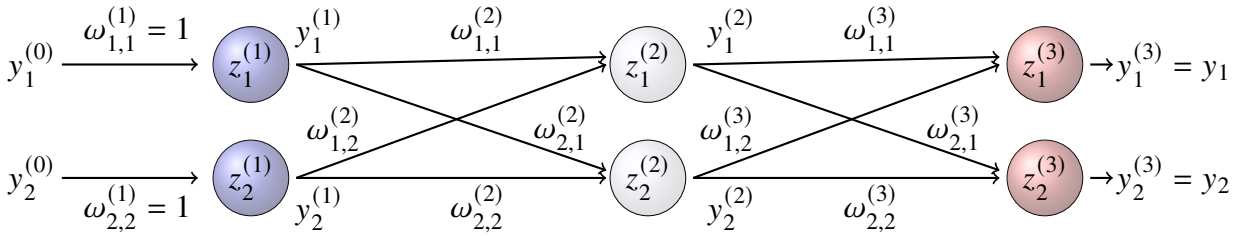
minden neuron esetén. Ennek az aktivációs függvénynek az a különlegessége, hogy a deriváltja:

$$\frac{\partial}{\partial z_j^{(i)}} l_j^{(i)}(z_j^{(i)}) = \frac{e^{-z_j^{(i)}}}{(1 + e^{-z_j^{(i)}})^2} = l_j^{(i)}(z_j^{(i)}) \cdot (1 - l_j^{(i)}(z_j^{(i)})) = y_j^{(i)} \cdot (1 - y_j^{(i)})$$

alakú, ezért $z_j^{(i)}$ értékek nélkül is kiszámolható. Hibafüggvényünk legyen a következő:

$$C(y, y') = \left\| y - y' \right\|^2 = (y_1 - y_1')^2 + (y_2 - y_2')^2$$

négyzetes hiba. A neurális hálózatunk pedig az alábbi struktúrájú:



4. ábra. Példa neurális hálózat.

Az $y^{(i)}$, $z^{(i)}$ vektorok adottak. Első lépésben számoljuk ki a hibafüggvény parciális deriváltjait a háló kimeneti változói szerint:

$$\frac{\partial}{\partial y_1} C(y_1, y_2, y_1', y_2') = \frac{\partial}{\partial y_1}((y_1 - y_1')^2 + (y_2 - y_2')^2) = 2(y_1 - y_1')$$

$$\frac{\partial}{\partial y_2} C(y_1, y_2, y_1', y_2') = \frac{\partial}{\partial y_2}((y_1 - y_1')^2 + (y_2 - y_2')^2) = 2(y_2 - y_2')$$

A korábban bemutatott rekurzió és a láncszabály alapján pedig kiszámolhatjuk a súlyok szerinti deriváltakat:

$$\frac{\partial}{\partial \omega_{u,v}^{(k)}} C(y_1, y_2, y_1', y_2') = \frac{\partial}{\partial y_1} C(y_1, y_2, y_1', y_2') \cdot \frac{\partial y_1}{\partial \omega_{u,v}^{(k)}} + \frac{\partial}{\partial y_2} C(y_1, y_2, y_1', y_2') \cdot \frac{\partial y_2}{\partial \omega_{u,v}^{(k)}}.$$

A 2. és 3. réteg közötti súlyok deriváljai:

$$\frac{\partial C}{\partial \omega_{1,1}^{(3)}} = 2(y_1^{(3)} - y_1')(1 - y_1^{(3)}) y_1^{(3)} y_1^{(2)},$$

$$\frac{\partial C}{\partial \omega_{1,2}^{(3)}} = 2(y_1^{(3)} - y_1')(1 - y_1^{(3)}) y_1^{(3)} y_2^{(2)},$$

$$\frac{\partial C}{\partial \omega_{2,1}^{(3)}} = 2(y_2^{(3)} - y_2')(1 - y_2^{(3)})y_2^{(3)}y_1^{(2)},$$
$$\frac{\partial C}{\partial \omega_{2,2}^{(3)}} = 2(y_2^{(3)} - y_2')(1 - y_2^{(3)})y_2^{(3)}y_2^{(2)}.$$

Az 1. és 2. réteg közötti súlyok deriváltjai:

$$\frac{\partial C}{\partial \omega_{1,1}^{(2)}} = \left(2(y_1^{(3)} - y_1')(1 - y_1^{(3)})y_1^{(3)}\omega_{1,1}^{(3)} + 2(y_2^{(3)} - y_2')(1 - y_2^{(3)})y_2^{(3)}\omega_{2,1}^{(3)}\right)(1 - y_1^{(2)})y_1^{(2)}y_1^{(1)},$$
$$\frac{\partial C}{\partial \omega_{1,2}^{(2)}} = \left(2(y_1^{(3)} - y_1')(1 - y_1^{(3)})y_1^{(3)}\omega_{1,1}^{(3)} + 2(y_2^{(3)} - y_2')(1 - y_2^{(3)})y_2^{(3)}\omega_{2,1}^{(3)}\right)(1 - y_1^{(2)})y_1^{(2)}y_2^{(1)},$$
$$\frac{\partial C}{\partial \omega_{2,1}^{(2)}} = \left(2(y_1^{(3)} - y_1')(1 - y_1^{(3)})y_1^{(3)}\omega_{1,2}^{(3)} + 2(y_2^{(3)} - y_2')(1 - y_2^{(3)})y_2^{(3)}\omega_{2,2}^{(3)}\right)(1 - y_2^{(2)})y_2^{(2)}y_1^{(1)},$$
$$\frac{\partial C}{\partial \omega_{2,2}^{(2)}} = \left(2(y_1^{(3)} - y_1')(1 - y_1^{(3)})y_1^{(3)}\omega_{1,2}^{(3)} + 2(y_2^{(3)} - y_2')(1 - y_2^{(3)})y_2^{(3)}\omega_{2,2}^{(3)}\right)(1 - y_2^{(2)})y_2^{(2)}y_2^{(1)}.$$

A gradiens módszer első fontos lépése tehát a gradiens kiszámolása. A backpropagation algoritmusban jelöljük a változókat az eddig definiált módon, továbbá legyen: $D_{i,j} = \frac{\partial C}{\partial y_j^{(i)}}$ és $G_{u,v}^{(k)} = \frac{\partial C}{\partial \omega_{u,v}^{(k)}}$. Y az $y_j^{(i)}$ értékeket, Z pedig a $z_j^{(i)}$ értékeket tartalmazó halmaz, továbbá A tartalmazza az aktivációs függvényeket ($l_j^{(i)}$).

1 Algoritmus: Backpropagation

Funct BACKPROP(Ω, Y, Z, A, C, d, r)

for $i = 1, ..., d(r)$ **do** ▷ elég d(r)-ig iterálni, mert d(r)+1 konstans nem érdekes
  $D_{r,i} := dC/dy_i^{(r)}$ ▷ kiszámoljuk a C hibafüggvény parciális deriváltjait
end for
for $i = r, ..., 1$ **do** ▷ itt a rétegeken iterálunk hátulról előre
  for $j = 1, ..., d(i-1) + 1$ **do** ▷ itt az $i-1$-edik réteg neuronjain iterálunk
    for $k = 1, .., d(i) + 1$ **do** ▷ itt az (i)-edik réteg neuronjain iterálunk
      if $k = 1$ **then** ▷ ha az első elemnél tartunk akkor nem adjuk hozzá a korábbi értékét
        $D_{i-1,j} := \omega_{k,j}^{(i)} \cdot dl_k^{(i)}/dz_k^{(i)} \cdot D_{i,k}$
      else
        $D_{i-1,j} := D_{i-1,j} + \omega_{k,j}^{(i)} \cdot dl_k^{(i)}/dz_k^{(i)} \cdot D_{i,k}$
      end if
    end for
  end for
  for $j = 1, ..., d(i-1) + 1$ **do** ▷ itt az $i-1$-edik réteg neuronjain iterálunk
    for $k = 1, .., d(i) + 1$ **do** ▷ itt az (i)-edik réteg neuronjain iterálunk
      $G_{k,j}^{(i)} := D_{i,k} \cdot y_j^{(i-1)}$
    end for
  end for
end for
return G ▷ visszatérünk a gradienssel

4. A batch gradiens módszer

4.1. A fix lépéshosszúságú gradiens módszer

A gradiens módszer, az az eljárás amellyel keressük egy adott $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ folytonosan differenciálható függvény valamely lokális minimumát, illetve minimumhelyét. Tekintsük az alábbi numerikus módszert:

$$x_{n+1} = x_n + \Delta v_n,$$
$$v_n = -\frac{\nabla g(x_n)}{\left\|\nabla g(x_n)\right\|},$$

ahol Δ a lépésközt jelöli, azaz egymást követően Δ hosszúságú lépést teszünk a minimális irányba. Az az intuíciónk, hogy ez az eljárás közelíteni fog valamely lokális minimumhelyhez. Nevezzük ezt a módszert fix lépéshosszú gradiens módszernek.

4.2. A görbe egyenletes konvergenciája

4.1. Állítás. Legyen $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ kétszer folytonosan differenciálható függvény, és tekintsünk benne egy $B(x_0, r)$ zárt hipergömböt, ahol $\nabla g(x_0) \neq 0$ Létezik olyan r, ahol $\forall x \in B(x_0, r)$ $\nabla g(x) \neq 0$, továbbá $\nabla g(x)$ Lipschitz tulajdonságú $B(x_0, r)$-ben.

4.1. Bizonyítás. $\nabla g(x)$ folytonos, ezért $(\forall \epsilon)$ $(\exists r)$ $(\forall x : \|x - x_0\| < r)$ $\left(\left\|\nabla g(x) - \nabla g(x_0)\right\| < \epsilon\right)$. $\epsilon = \left\|\nabla g(x_0)\right\|$ választással adódik megfelelő r. Mivel $g^{(2)}(x)$ folytonos, ezért $(\nabla g(x))'$ is folytonos. $B(x_0, r)$ zárt halmaz, így a Weierstrass-tétel miatt $\left\|\nabla g(x)'\right\|$ korlátos $B(x_0, r)$-en, és fel is veszi maximumát. A Lagrange egyenlőtlenség miatt $\forall x, y \in B(x_0, r)$:

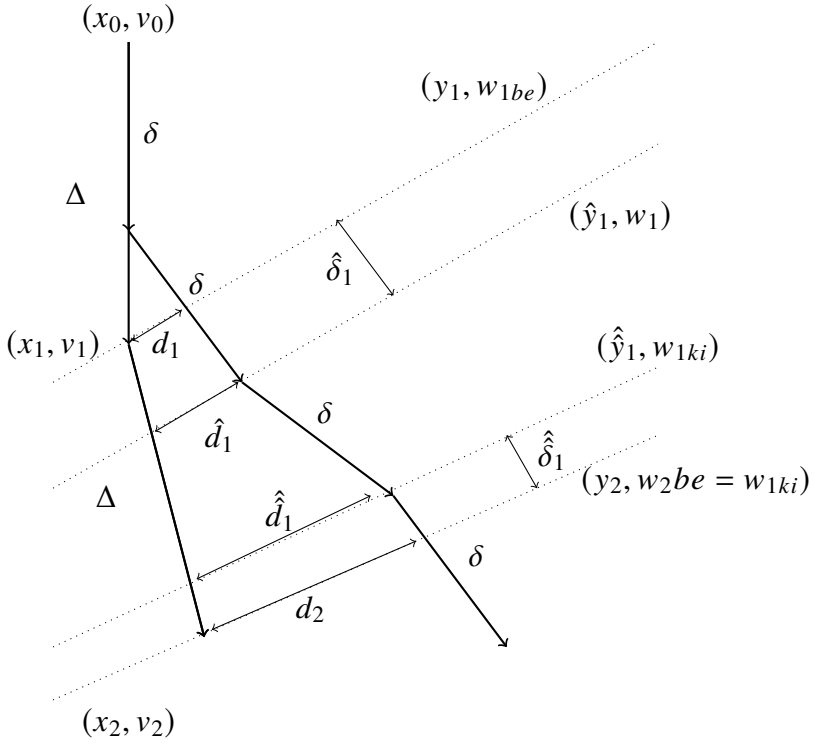
$$\left\|\nabla g(x) - \nabla g(y)\right\| \leq \sup_{z \in s(x,y)} \left\{\left\|(\nabla g(z))'(x - y)\right\|\right\} \leq \sup_{z \in s(x,y)} \left\{\left\|(\nabla g(z))'\right\|\right\}\left\|x - y\right\| \leq$$

$$\leq \sup_{z \in B(x_0,r)} \left\{\left\|(\nabla g(z))'\right\|\right\}\left\|x - y\right\|,$$

tehát $L = \sup_{z \in B(x_0,r)} \left\{\left\|(\nabla g(z))'\right\|\right\}$ érték megfelel a Lipschitz tulajdonsághoz. □

4.2. Állítás. Jelölje $\gamma_n^{(l)}(t) : \mathbb{R} \longrightarrow \mathbb{R}^d$ azt a folytonos töröttvonalat, melyet a fix lépéshosszú gradiens módszer által kapot x_i pontokat köti össze a $g : \mathbb{R}^d \longrightarrow \mathbb{R}$ kétszer folytonosan differenciálható függvényre alkalmazva, ahol $\Delta = l/n$. Legyen ez a görbe ívhossz szerinti paraméterezésű, továbbá $\nabla g(x_0) \neq 0$ és $x \in B(x_0, r) : \nabla g(x) \neq 0$. Ekkor $\left(\gamma_n^{(l)}(t)\right)_{n\in N}$, $l < r$ függvénysorozat egyenletesen konvergens.

4.2. Bizonyítás. Első lépésben nézzük meg egy Δ és egy $\delta \in [\frac{\Delta}{2}, \Delta]$ iteráció viszonyát. Az ábrán (5. ábra) x_i és v_i jelöli a Δ iteráció töréspontjait és irányait, ehhez hasonlóan $y_i, \hat{y}_i, \hat{\hat{y}}_i$ és w_{ibe}, w_i, w_{iki} a δ iteráció töréspontjait és irányait. Utóbbiból azért kell több, mert egy Δ iterációs lépésben várhatónan 2 δ iterációs lépés történik. $d_i, \hat{d}_i, \hat{\hat{d}}_i$ pedig minden töréspontnál méri a távolságot a két iteráció között.



5. ábra. Itererációs lépések.

Határozzuk meg a töréspontoknál lévő távolságok változását egy Δ iteráció lefutása alatt. Ehhez használjuk fel a Lipschitz tulajdonságot és a háromszög egyenlőtlenséget:

$$\hat{d}_i \leq \left\| (x_i + \hat{\delta}_i v_i) - (y_i + \hat{\delta} w_i) \right\|,$$
$$\hat{d}_i \leq \left\| (x_i - y_i) + \hat{\delta}_i (v_i - w_i) \right\|,$$
$$\hat{d}_i \leq \left\| x_i - y_i \right\| + \left\| \hat{\delta}_i (v_i - w_i) \right\|,$$
$$\hat{d}_i \leq d_i + \hat{\delta}_i L d_i,$$
$$\hat{d}_i \leq (\hat{\delta}_i L + 1) d_i.$$

Hasonló módon adódik a többi távolság is:

$$\hat{\hat{d}}_i \leq (\delta L + 1)\hat{d}_i,$$

$$d_{i+1} \leq (\hat{\hat{\delta}}_i L + 1)\hat{\hat{d}}_i.$$

Azaz egy Δ iteráció alatt így lehet felülről becsülni a távolság változását:

$$d_{i+1} \leq (\hat{\delta}_i L + 1)(\delta L + 1)(\hat{\hat{\delta}}_i L + 1)d_i.$$

Mivel feltettük, hogy $\delta \in [\frac{\Delta}{2}, \Delta]$ ezért majorálhatjuk a kifejezést az alábbi módon:

$$d_{i+1} \leq (\Delta L + 1)^3 d_i.$$

Az első Δ iteráció felültől becsülhető ΔL-el így:

$$d_i \leq d_1(\Delta L + 1)^{3(i-1)},$$

,

$$d_i \leq \Delta L(\Delta L + 1)^{3(i-1)}.$$

$\Delta = l/n$ alapján pedig felírhatjuk a felosztás függvényében a két iteráció maximális távolságát:

$$d_n \leq \frac{l}{n} L(\frac{l}{n} L + 1)^{3(n-1)},$$

$$d_n \leq \frac{l}{n} L\left((\frac{1}{\frac{n}{lL}} + 1)^{\frac{n}{lL}}\right)^{3lL}\left(\frac{l}{n} L + 1\right)^{-3},$$

$$d_n \leq \frac{l}{n} L e^{3lL}.$$

A sorozat elemeit particionáljuk úgy, hogy ha $n \in [2^k, 2^{k+1})$ akkor n a k-ik partícióba kerüljön. Ekkor a háromszög egyenlőtlenségből adódóan a j-edik parcítió után, két iteráció maximális távolsága:

$$lLe^{3lL} \sum_{k=j}^{\infty} \frac{1}{2^k},$$

$$lLe^{3lL} \frac{1}{2^{k-1}}.$$

Továbbá:

$$\lim_{k \longrightarrow \infty} lLe^{3lL} \frac{1}{2^{k-1}} = 0,$$

azaz a függvénysorozat egyenletesen Cauchy-tulajdonságú, tehát egyenletesen konvergens. Ezzel az állítást beláttuk. □

4.1. Következmény. Mivel $\gamma_n^{(l)}(t)$ folytonos $\forall n$-re és $\left(\gamma_n^{(l)}(t)\right)_{n\in N} \longrightarrow \gamma^{(l)}(t)$ egyenletesen, ezért $\gamma^{(l)}(t)$ folytonos.

4.3. Állítás. A $\gamma^{(l)}(t)$ görbe folytonosan differenciálható, és $(\gamma^{(l)}(t))' = -\frac{\nabla g(\gamma^{(l)}(t))}{\|\nabla g(\gamma(l)(t))\|_2}$.

4.3. Bizonyítás. Először vizsgáljuk meg a $\gamma_n^{(l)}(t_0+T) - \gamma_n^{(l)}(t_0)$ különbséget olyan iterációkban, ahol $\Delta < T$ tehát az iterációs lépésünk már kisebb mint a két pont közötti görbehossz. Nem biztos, hogy az adott iterációban a kezdőpont töréspont is, hanem a töréspont $t_0 - \delta$ értéknél van, illetve az utolsó lépés nem feltétlenül Δ hosszúságú. Ezt jelöljük δ'-vel. A lépések száma legyen $n+2$, ahol 1 darab $\Delta - \delta$ hosszúságú kezdőlépést, n darab Δ hosszúságú általános lépést, majd 1 darab δ' hosszúságú befejező lépést teszünk.
A $\gamma_n^{(l)}(t_0)$ pontból, ha az töréspont, akkor az irány amivel lépünk először az iterációban:

$$v_{n,t_0} = -\frac{\nabla g(\gamma_n^{(l)}(t_0))}{\left\|\nabla g(\gamma_n^{(l)}(t_0))\right\|_2}.$$

Ha $\gamma_n^{(l)}(t_0)$ nem töréspont, akkor az előző töréspontból számolt irányba lépünk, csak kisebb távolságot az első lépésben. Ekkor:

$$v_{n,t_0} = -\frac{\nabla g(\gamma_n^{(l)}(t_0-\delta))}{\left\|\nabla g(\gamma_n^{(l)}(t_0-\delta))\right\|_2}.$$

A háromszög egyenlőtlenség miatt minden lépésben a kiszámolt irányra igaz, hogy:

$$\left\|v_{n,(t_0+\Delta-\delta+k\Delta)} - v_{n,t_0}\right\|_2 \leq (\Delta - \delta + k\Delta)L.$$

Ebből következik, hogy minden koordinátára teljesül a következő:

$$(v_{n,(t_0+\Delta-\delta+k\Delta),i} - v_{n,t_0,i})^2 \leq ((\Delta - \delta + k\Delta)L)^2,$$

$$|v_{n,(t_0+\Delta-\delta+k\Delta),i} - v_{n,t_0,i}| \leq (\Delta - \delta + k\Delta)L.$$

Ebből következik, hogy:

$$v_{n,t_0,i} - (\Delta - \delta + k\Delta)L \leq v_{n,(t_0+\Delta-\delta+k\Delta),i} \leq v_{n,t_0,i} + (\Delta - \delta + k\Delta)L.$$

Ezek után felírhatjuk az egész vektorra, hogy:

$$v_{n,t_0} - (\Delta - \delta + k\Delta)L \cdot \mathbb{1} \leq v_{n,(t_0+\Delta-\delta+k\Delta)} \leq v_{n,t_0} + (\Delta - \delta + k\Delta)L \cdot \mathbb{1}.$$

A $(\gamma_n^{(l)}(t_0+T) - \gamma_n^{(l)}(t_0))$ különbség így írható fel ($(\Delta-\delta)$, (δ') az első és utolsó nem feltétlenül teljes hosszú lépés, (n) a teljes (Δ) hosszú lépések száma):

$$\gamma_n^{(l)}(t_0+T) - \gamma_n^{(l)}(t_0) = v_{n,t_0}(\Delta - \delta) + \left(\sum_{k=1}^{n} v_{n,(t_0+\Delta-\delta+k\Delta)}\Delta\right) + v_{n,(t_0+\Delta-\delta+(n+1)\Delta)}\delta'.$$

Becsüljük alulról és felülről a szummában lévő részt, az irányvektorokra adott egyenlőtlenség összeadásával:

$$\sum_{k=1}^{n}(v_{n,t_0} - (\Delta - \delta + k\Delta)L \cdot \mathbb{1})\Delta \leq \sum_{k=1}^{n} v_{n,(t_0+\Delta-\delta+k\Delta)}\Delta \leq \sum_{k=1}^{n}(v_{n,t_0} + (\Delta - \delta + k\Delta)L \cdot \mathbb{1})\Delta,$$

melyből az alábbi alsó és felső becslés adódik:

$$n\Delta v_{n,t_0} - (nL\Delta^2 - nL\delta\Delta + \frac{n(n+1)}{2}L\Delta^2)\mathbb{1} \leq \sum_{k=1}^{n} v_{n,(t_0+\Delta-\delta+k\Delta)}\Delta,$$

$$\sum_{k=1}^{n} v_{n,(t_0+\Delta-\delta+k\Delta)}\Delta \leq n\Delta v_{n,t_0} + (nL\Delta^2 - nL\delta\Delta + \frac{n(n+1)}{2}L\Delta^2)\mathbb{1}.$$

Mindezek alapján tudjuk alulról és felülről becsülni $\Delta\gamma_n^{(l)}(t_0) = (\gamma_n^{(l)}(t_0+T) - \gamma_n^{(l)}(t_0))$ értéket. Felírjuk a kezdő lépést, a szummában lévő Δ lépések becslését, illetve az utolsó lépés (δ') becslését is:

$$v_{n,t_0}(\Delta-\delta)+n\Delta v_{n,t_0}-(nL\Delta^2-nL\delta\Delta+\frac{n(n+1)}{2}L\Delta^2)\mathbb{1}+v_{n,t_0}\delta'-(\Delta-\delta+(n+1)\Delta)L\delta'\mathbb{1} \leq \Delta\gamma_n^{(l)}(t_0),$$

$$\Delta\gamma_n^{(l)}(t_0) \leq v_{n,t_0}(\Delta-\delta)+n\Delta v_{n,t_0}+(nL\Delta^2-nL\delta\Delta+\frac{n(n+1)}{2}L\Delta^2)\mathbb{1}+v_{n,t_0}\delta'+(\Delta-\delta+(n+1)\Delta)L\delta'\mathbb{1}.$$

Tekintsük most csak a felső becslést, az alsó becslés ezzel megegyezik annyi különbséggel, hogy ott az $\mathbb{1}$ vektor együtthatója ellentétes előjelű:

$$\Delta\gamma_n^{(l)}(t_0) \leq v_{n,t_0}(\Delta - \delta + n\Delta + \delta') + (nL\Delta^2 - nL\delta\Delta + \frac{n(n+1)}{2}L\Delta^2 + (\Delta - \delta + (n+1)\Delta)L\delta')\mathbb{1}.$$

Vegyük észre hogy: $\Delta - \delta + n\Delta + \delta' = T$, továbbá vezessük be $T' = n\Delta$ jelölést. Ekkor az alábbi egyenlethez jutunk algebrai átalakítások után:

$$\Delta\gamma_n^{(l)}(t_0) \leq v_{n,t_0}T + \Big(\frac{1}{2}L(T')^2 + L(\frac{3}{2}\Delta - \delta + \delta')T' + L\delta'(2\Delta - \delta)\Big)\mathbb{1}.$$

Tudván, hogy $\delta < \Delta$, $\delta' < \Delta$ és $T' < T$, ezt felülről becsülhetjük, (itt most felírjuk az alsó becslést is ennek megfelelően):

$$v_{n,t_0}T - L\Big(\frac{1}{2}T^2 + (\frac{5}{2}T + 2\Delta)\Delta\Big)\mathbb{1} \leq \Delta\gamma_n^{(l)}(t_0) \leq v_{n,t_0}T + L\Big(\frac{1}{2}T^2 + (\frac{5}{2}T + 2\Delta)\Delta\Big)\mathbb{1}.$$

Tudjuk, hogy $v_{n,t_0} = -\frac{\nabla g(\gamma_n^{(l)}(t_0-\delta))}{\left\|\nabla g(\gamma_n^{(l)}(t-\delta))\right\|_2}$. Mivel $\nabla g(x)$ folytonos és $\gamma(t)$ is folytonos akkor $-\frac{\nabla g(\gamma_n^{(l)}(t))}{\left\|\nabla g(\gamma_n^{(l)}(t)\right\|_2}$ is folytonos. Felhasználva, hogy $\gamma_n^{(l)}(t_0) \longrightarrow \gamma^{(l)}(t_0)$ és $\Delta \longrightarrow 0$ miatt $\delta \longrightarrow 0$ kapjuk, hogy:

$$\lim_{n\longrightarrow\infty} v_{n,t_0} = \lim_{n\longrightarrow\infty} = -\frac{\nabla g(\gamma_n^{(l)}(t_0-\delta))}{\left\|\nabla g(\gamma_n^{(l)}(t_0-\delta))\right\|_2} = -\frac{\nabla g(\gamma^{(l)}(t_0))}{\left\|\nabla g(\gamma^{(l)}(t_0))\right\|_2} = v_{t_0}.$$

Továbbá $\gamma_n^{(l)}(t_0+T) \longrightarrow \gamma^{(l)}(t_0+T)$ és:

$$\gamma_n^{(l)}(t_0)+v_{n,t_0}T-L\Big(\frac{1}{2}T^2+(\frac{5}{2}T+2\Delta)\Delta\Big)\mathbb{1} \leq \gamma_n^{(l)}(t_0+T) \leq \gamma_n^{(l)}(t_0)+v_{n,t_0}T+L\Big(\frac{1}{2}T^2+(\frac{5}{2}T+2\Delta)\Delta\Big)\mathbb{1}.$$

Vegyük a határértékeket $n \longrightarrow \infty$ esetén, amikor $\Delta \longrightarrow 0$ teljesül:

$$\gamma^{(l)}(t_0) + v_{t_0}T - L\frac{1}{2}T^2\mathbb{1} \leq \gamma^{(l)}(t_0+T) \leq \gamma^{(l)}(t_0) + v_{t_0}T + L\frac{1}{2}T^2\mathbb{1}.$$

Ezekből felírhatjuk a differenciahányadost:

$$\frac{v_{t_0}T - L\frac{1}{2}T^2\mathbb{1}}{T} \leq \frac{\gamma(t_0+T)-\gamma(t_0)}{T} \leq \frac{v_{t_0}T + L\frac{1}{2}T^2\mathbb{1}}{T}.$$

Vagyis $T \longrightarrow 0$ határértéket véve:

$$\lim_{T\longrightarrow 0} v_{t_0} - L\frac{1}{2}T\mathbb{1} \leq (\gamma^{(l)}(t_0))'_+ \leq \lim_{T\longrightarrow 0} v_{t_0} + L\frac{1}{2}T\mathbb{1},$$

$$-\frac{\nabla g(\gamma^{(l)}(t_0))}{\left\|\nabla g(\gamma^{(l)}(t_0))\right\|_2} = v_{t_0} \leq (\gamma^{(l)}(t_0))'_+ \leq v_{t_0} = -\frac{\nabla g(\gamma^{(l)}(t_0))}{\left\|\nabla g(\gamma^{(l)}(t_0))\right\|_2},$$

$$(\gamma^{(l)}(t_0))'_+ = -\frac{\nabla g(\gamma^{(l)}(t_0))}{\left\|\nabla g(\gamma^{(l)}(t_0))\right\|_2}.$$

Az bizonyítást alkalmazhatjuk hasonlóképp $(\gamma^{(l)}(t_0))'_-$-ra is, ezért:

$$(\gamma^{(l)}(t_0))'_- = (\gamma^{(l)}(t_0))'_+ = (\gamma^{(l)}(t_0))' = -\frac{\nabla g(\gamma^{(l)}(t_0))}{\left\|\nabla g(\gamma^{(l)}(t_0))\right\|_2}.$$

Mivel $-\frac{\nabla g(\gamma^{(l)}(t))}{\left\|\nabla g(\gamma^{(l)}(t))\right\|_2}$ folytonos, ezért $\gamma^{(l)}(t)$ folytonosan differenciálható is. Ezzel az állítást beláttuk. □

4.2. Következmény. $\gamma^{(l)}(t)$ görbe megoldása az alábbi differenciálegyenletnek:

$$t \in [0,l], \qquad \gamma^{(l)}(0) = x_0, \qquad g(\gamma^{(l)}(t))'_t = -\frac{\nabla g(\gamma^{(l)}(t))}{\left\|\nabla g(\gamma^{(l)}(t))\right\|_2}.$$

Sikerült megmutatnunk, hogy a numerikus módszerünk, egy olyan görbét közelít, mely minden pontjában a legjobban csökkenő irányba mutat. Intuitíve láthatjuk, hogy ez a görbe alkalmas lokális minimumok meghatározásához, ezért az iterációnk is alkalmas erre. A későbbi fejezetekben ennek precíz igazolását láthatjuk majd.

4.3. A változó lépéshosszúságú gradiens módszer

Az iteráció során érdemes nem állandó lépéshosszúságot választanunk. Intuitíve ennek az az oka, hogy az optimumtól távol, haladhatunk nagyobb lépésekben, ellenben ahogy közelítünk az optimális megoldás felé, egyre inkább finomítanunk kell a lépések hosszát, hogy minél pontosabb megoldást kapjunk.

Legyen $g : \mathbb{R}^n \longrightarrow \mathbb{R}$ folytonosan differenciálható függvény és tekintsük az alábbi numerikus módszert:

$$x_{k+1} = x_k + \alpha_k v_k$$
$$v_k = -\nabla G(x_k)$$

ahol α_k a lépéshosszok sorozata, illetve v_k az irány, amelybe lépünk az iterációban. Mivel a lépéshosszok változnak, így v_k normálására nincs szükségünk. Nevezzük ezt a módszert változó lépéshosszúságú gradiens módszernek.
Ennek a módszernek a konvergenciája az optimális megoldáshoz a sztochasztikus gradiens módszer konvergenciájának egy speciális esete, így ebben a fejezetben erre külön nem térünk ki.

5. A sztochasztikus gradiens módszer

5.1. A módszer bemutatása

Tekintsük az alábbi optimalizációs feladatot ahol $g_i : R^d \longrightarrow R$ és $G : R^d \longrightarrow R$:

$$min\{G(x) = \frac{1}{n}\sum_{i=1}^{n} g_i(x) : x \in R^d\},$$

azaz szeretnénk minimalizálni a $g_i(x)$ függvények tapasztalati várható értékét. A batch gradiens módszer alkalmazása esetén $G(x)$ gradiense az alábbi alakban írható fel:

$$\nabla G(x) = \frac{1}{n}\sum_{i=1}^{n} \nabla g_i(x).$$

Abban az esetben, ha d és n nagy érték, akkor ennek a kiszámolása nagyon nagy számítási költséggel jár. A sztochasztikus gradiens alapötlete az, hogy ne vegyük figyelembe a gradiens számolásánál az összes $g_i(x)$ tagot, hanem csak nézzünk egy $S \subseteq \{1, 2, ..., n\}$ részhalmazt és csak ezeket használjuk fel a gradiens kiszámolása során. Tekintsük ennek a részhalmaznak a tapasztalati várható értékét, és ennek gradiensét:

$$\xi \in \{0, 1\}^n,$$

$$\xi_i = \mathbb{1}_{i \in S},$$

$$G_\xi(x) = \frac{1}{\|\xi\|_1}\sum_{i=1}^{n} g_i(x)\xi_i,$$

$$g(x, \xi) = \nabla G_\xi(x) = \frac{1}{\|\xi\|_1}\sum_{i=1}^{n} \nabla g_i(x)\xi_i.$$

Legyen α_k a lépéshosszok sorozata és ξ_k pedig random változók sorozata. Az alábbi iterációt:

$$x_{k+1} = x_k + v_k\alpha_k$$

$$v_k = -g(x_k, \xi_k)$$

sztochasztikus gradiens módszernek nevezzük.

5.2. Alapvető tulajdonságok

5.1. Lemma. Legyen $G : \mathbb{R}^d \longrightarrow \mathbb{R}$, folytonosan differenciálható függvény, továbbá, $\nabla G(x)$ Lipschitz-folytonos. Ekkor teljesül az alábbi egyenlőtlenség $\forall x, y \in R^d$ esetén:

$$G(y) \leq G(x) + \nabla G(x)^T (y - x) + \frac{1}{2} L \|x - y\|_2^2 .$$

5.1. Bizonyítás. Írjuk fel $G(y)$ értéket a Newton-Leibnitz-tétellel:

$$G(y) = G(x) + \int_0^1 \frac{\partial G(x + t(y - x))}{\partial t} dt = G(x) + \int_0^1 \nabla G(x + t(y - x))^T (x - y) dt =$$

$$= G(x) + \nabla G(x)^T (y - x) + \int_0^1 (\nabla G(x + t(y - x))^T - \nabla G(x)^T)(y - x) dt \leq$$

$$\leq G(x) + \nabla G(x)^T (y - x) + \int_0^1 Lt \|y - x\|_2 \|y - x\|_2 \, dt = G(x) + \nabla G(x)^T (y - x) + \frac{1}{2} L \|y - x\|_2^2 . \square$$

A továbbiakban feltesszük, hogy $\nabla G(x)$ Lipschitz-folytonos.

5.2. Lemma. A sztochasztikus gradiens módszer teljesíti az alábbi egyenlőtlenséget:

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - G(x_k) \leq -\alpha_k \nabla G(x_k)^T \mathbb{E}_{\xi_k}[g(x_k, \xi_k)] + \frac{1}{2} \alpha_k^2 L \mathbb{E}_{\xi_k}[\|g(x_k, \xi_k)\|_2^2].$$

5.2. Bizonyítás. Felhasználva az 5.1 Lemma állítását:

$$G(x_{k+1}) - G(x_k) \leq \nabla G(x_k)^T (x_{k+1} - x_k) + \frac{1}{2} L \|x_{k+1} - x_k\|_2^2 \leq$$

$$\leq -\alpha_k \nabla G(x_k)^T g(x_k, \xi_k) + \frac{1}{2} \alpha_k^2 L \|g(x_k, \xi_k)\|_2^2 ,$$

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - G(x_k) \leq -\alpha_k \nabla G(x_k)^T \mathbb{E}_{\xi_k}[g(x_k, \xi_k)] + \frac{1}{2} \alpha_k^2 L \mathbb{E}_{\xi_k}[\|g(x_k, \xi_k)\|_2^2].$$

Ezzel az állítást beláttuk. □

Azért, hogy a konvergenciát biztosítsuk, be kell vezetünk kritériumokat $g(x_k, \xi_k)$ varianciájának értékére. Definiáljuk a varianciát az alábbi módon:

$$\mathbb{V}_{\xi_k}[g(x_k, \xi_k)] := \mathbb{E}_{\xi_k}[\|g(x_k, \xi_k)\|_2^2] - \|\mathbb{E}_{\xi_k}[g(x_k, \xi_k)]\|_2^2 .$$

Ahhoz, hogy állításokat tudjunk megfogalmazni az iteráció konvergenciájával összefüggésben, bevezetünk három feltételt. Tegyük fel, hogy az iterációban $\forall x_k \in H$, ahol H nyílt halmaz továbbá H alulról korlátos, azaz $\exists G_{inf} : \forall x \in H$ esetén $G_{inf} \leq x$. Továbbá létezik $\mu_G > \mu > 0$ és $M \geq 0$, $M_v \geq 0$, hogy $\forall k \in \mathbb{N}$ esetén teljesülnek az alábbi feltételek:

(A1): $\nabla G(x_k)^T \mathbb{E}_{\xi_k}[g(x_k, \xi_k)] \geq \mu \|\nabla G(x_k)\|_2^2$

(A2): $\|\mathbb{E}_{\xi_k}[g(x_k, \xi_k)]\|_2 \leq \mu_G \|\nabla G(x_k)\|_2$

(A3): $\mathbb{V}[g(x_k, \xi_k)] \leq M + M_V \|\nabla G(x_k)\|_2^2$

5.1. Állítás. Az $(A1) \longrightarrow (A3)$ feltételek mellett alkalmas $M \geq 0$, $M_G \geq 0$ esetén igaz az alábbi egyenlőtlenség:

$$\mathbb{E}_{\xi_k}[\left\|g(x_k, \xi_k)\right\|_2^2] \leq M + M_G\left\|\nabla G(x_k)\right\|.$$

5.3. Bizonyítás. Először használjuk fel $(A2)$-as feltételt:

$$\mathbb{V}[g(x_k, \xi_k)] = \mathbb{E}_{\xi_k}[\left\|g(x_k, \xi_k)\right\|_2^2] - \left\|\mathbb{E}_{\xi_k}[g(x_k, \xi_k)]\right\|_2^2 \geq \mathbb{E}_{\xi_k}[\left\|g(x_k, \xi_k)\right\|_2^2] - \mu_G^2\left\|\nabla G(x_k)\right\|_2^2.$$

Majd erre alkalmazzuk az $(A3)$-as feltételt:

$$\mathbb{E}_{\xi_k}[\left\|g(x_k, \xi_k)\right\|_2^2] - \mu_G^2\left\|\nabla G(x_k)\right\|_2^2 \leq \mathbb{V}[g(x_k, \xi_k)] \leq M + M_V\left\|\nabla G(x_k)\right\|_2^2,$$

$$\mathbb{E}_{\xi_k}[\left\|g(x_k, \xi_k)\right\|_2^2] - \mu_G^2\left\|\nabla G(x_k)\right\|_2^2 \leq M + M_V\left\|\nabla G(x_k)\right\|_2^2.$$

Ezekből következik hogy:

$$\mathbb{E}_{\xi_k}[\left\|g(x_k, \xi_k)\right\|_2^2] \leq M + M_G\left\|\nabla G(x_k)\right\|_2^2,$$

ahol:

$$0 < \mu^2 \leq M_V + \mu_G^2 = M_G.\square$$

5.3. Lemma. Az $(A1) \longrightarrow (A3)$ feltételek mellett teljesülnek az alábbi egyenlőtlenségek:

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - G(x_k) \leq -\mu\alpha_k\left\|\nabla G(x_k)\right\|_2^2 + \frac{1}{2}\alpha_k^2 L\mathbb{E}_{\xi_k}[\left\|g(x_k, \xi_k)\right\|_2^2],$$

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - G(x_k) \leq -(\mu - \frac{1}{2}\alpha_k LM_G)\alpha_k\left\|\nabla G(x_k)\right\|_2^2 + \frac{1}{2}\alpha_k^2 LM.$$

5.4. Bizonyítás. Felhasználva az 5.2-es Lemmát, illetve az $(A1)$ feltételt:

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - G(x_k) \leq -\alpha_k\nabla G(x_k)^T\mathbb{E}_{\xi_k}[g(x_k, \xi_k)] + \frac{1}{2}\alpha_k^2 L\mathbb{E}_{\xi_k}[\left\|g(x_k, \xi_k)\right\|_2^2],$$

$$\nabla G(x_k)^T\mathbb{E}_{\xi_k}[g(x_k, \xi_k)] \geq \mu\left\|\nabla G(x_k)\right\|_2^2,$$

amelyből adódik, hogy:

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - G(x_k) \leq -\mu\alpha_k\left\|\nabla G(x_k)\right\|_2^2 + \frac{1}{2}\alpha_k^2 L\mathbb{E}_{\xi_k}[\left\|g(x_k, \xi_k)\right\|_2^2].$$

Majd használjuk fel az 5.1-es Állítást, amelyből már egyszerűen kapjuk hogy:

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - G(x_k) \leq -(\mu - \frac{1}{2}\alpha_k LM_G)\alpha_k\left\|\nabla G(x_k)\right\|_2^2 + \frac{1}{2}\alpha_k^2 LM.$$

A Lemmát beláttuk. □

5.3. A sztochasztikus gradiens vizsgálata erősen konvex függvények esetén

Ebben a fejezetben megvizsgáljuk a sztochasztikus gradiens módszer konvergenciáját erősen konvex függvények esetében.

5.1. Definíció. Legyen $G : \mathbb{R}^d \longrightarrow \mathbb{R}$ folytonosan differenciálható. $G(x)$ erősen konvex függvény, ha létezik $c > 0$ amire:

$$G(y) \geq G(x) + \nabla G(x)^T (y - x) + \frac{1}{2} c \|x - y\|_2^2$$

egyenlőség teljesül.
Először bebizonyítjuk az alábbi lemmát, amit több bizonyításban fogunk a fenti konstans c esetén használni.

5.4. Lemma. Legyen G_* a $G(x)$ függvény minimumhelye. Ekkor teljesül az alábbi egyenlőtlenség $\forall x \in R^d$ esetén:

$$2c(G(x) - G_*) \leq \|\nabla G(x)\|_2^2 .$$

5.5. Bizonyítás. Legyen $q(y) = G(x) + \nabla G(x)^T (y - x) + \frac{1}{2} c \|y - x\|_2^2$. A $q(y)$ függvény minimumhelye adott x esetén $\hat{y}_* := x - \frac{1}{c} \nabla G(x)$ hiszen:

$$q'(y) = \nabla G(x)^T + c(y - x) = \nabla G(x)^T + cy - cx = 0$$

esetén

$$x - \frac{1}{c} \nabla G(x) = y.$$

Továbbá:

$$q''(y) = cI,$$

ahol $c > 0$ miatt $q''(y)$ pozitív definit, így $\hat{y}_* = x - \frac{1}{c} \nabla G(x)$ valóban minimumhely
$G(x) - \frac{1}{2c} \|\nabla G(x)\|_2^2$ értékkel.
Legyen y_* G minimumhelye, azaz $G(y_*) = G_*$. Ekkor G erősen konvex tulajdonsága miatt:

$$G_* \geq G(x) + \nabla G(x)^T (y_* - x) + \frac{1}{2} c \|x - y_*\|_2^2 \geq G(x) - \frac{1}{2c} \|\nabla G(x)\|_2^2 ,$$

amiből következik, hogy:

$$2c(G(x) - G_*) \leq \|\nabla G(x)\|_2^2 .$$

□

Ebben a részben bebizonyítjuk az első konvergenciatételt a sztochasztikus gradiensről, fix lépéshossz esetén. Intuitíve láthatjuk, hogy ebben az esetben konvergencia az optimum egy adott környezetére érvényes, azaz valamilyen ϵ sugáron belül leszünk az optimális megoldástól.

Vezessük be az alábbi jelölést:

$$\mathbb{E}[G(x_k)] = \mathbb{E}_{\xi_1}\mathbb{E}_{\xi_2}...\mathbb{E}_{\xi_{k-1}}[G(x_k)].$$

5.1. Tétel. Legyen $G(x) : \mathbb{R}^d \longrightarrow \mathbb{R}$ továbbá $\nabla G(x)$ Lipschitz-folytonos, illetve teljesüljenek az $(A1) \longrightarrow (A3)$ feltételek a sztochasztikus gradiens iterációs pontjaira. A lépéshossz $\alpha_k = \hat{\alpha}$ legyen fix, és teljesüljön az alábbi feltétel:

$$0 < \hat{\alpha} < \frac{\mu}{LM_G}.$$

Ekkor a várható eltérés az optimumtól:

$$\mathbb{E}[G(x_k) - G_*] \leq \frac{\hat{\alpha}LM}{2c\mu} + (1 - \hat{\alpha}c\mu)^{k-1}\Big(G(x_1) - G_* - \frac{\hat{\alpha}LM}{2c\mu}\Big),$$

$$\lim_{k\longrightarrow\infty} \mathbb{E}[G(x_k) - G_*] \leq \frac{\hat{\alpha}LM}{2c\mu}.$$

5.6. Bizonyítás. Használjuk fel az 5.3-as és 5.4-es Lemmákat, illetve a lépéshosszról szóló feltételt:

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - G(x_k) \leq -(\mu - \frac{1}{2}\hat{\alpha}LM_G)\hat{\alpha}\big\|\nabla G(x_k)\big\|_2^2 + \frac{1}{2}\hat{\alpha}^2LM,$$

$$2c(G(x) - G_*) \leq \nabla\big\|G(x)\big\|_2^2.$$

Először használjuk fel az 5.3-es Lemmát és a lépéshosszról szóló feltételt:

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - G(x_k) \leq -(\mu - \frac{1}{2}\hat{\alpha}LM_G)\hat{\alpha}\big\|\nabla G(x_k)\big\|_2^2 + \frac{1}{2}\hat{\alpha}^2LM \leq$$

$$\leq -(\mu - \frac{1}{2}\frac{\mu}{LM_G}LM_G)\hat{\alpha}\big\|\nabla G(x_k)\big\|_2^2 + \frac{1}{2}\hat{\alpha}^2LM =$$

$$= -\frac{1}{2}\mu\hat{\alpha}\big\|\nabla G(x_k)\big\|_2^2 + \frac{1}{2}\hat{\alpha}^2LM.$$

Most az 5.4-es Lemma alapján:

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - G(x_k) \leq -\frac{1}{2}\mu\hat{\alpha}\big\|\nabla G(x_k)\big\|_2^2 + \frac{1}{2}\hat{\alpha}^2LM \leq$$

$$\leq -\frac{1}{2}\mu\hat{\alpha}2c(G(x_k) - G_*) + \frac{1}{2}\hat{\alpha}^2LM = -\mu\hat{\alpha}c(G(x_k) - G_*) + \frac{1}{2}\hat{\alpha}^2LM.$$

Vonjunk le mindjét oldalból G_*-ot majd átrendezés után vegyük a teljes várható értéket:

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - G(x_k) - G_* \leq -G_* - \mu\hat{\alpha}c(G(x_k) - G_*) + \frac{1}{2}\hat{\alpha}^2LM,$$

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - G_* \leq G(x_k) - G_* - \mu\hat{\alpha}c(G(x_k) - G_*) + \frac{1}{2}\hat{\alpha}^2LM,$$

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - G_* \leq (1 - \mu\hat{\alpha}c)(G(x_k) - G_*) + \frac{1}{2}\hat{\alpha}^2 LM,$$

$$\mathbb{E}[G(x_{k+1}) - G_*] \leq (1 - \mu\hat{\alpha}c)\mathbb{E}[G(x_k) - G_*] + \frac{1}{2}\hat{\alpha}^2 LM.$$

Vonjunk le mindkét oldalból $\frac{\alpha\hat{L}M}{2c\mu}$ konstanst, amiből:

$$\mathbb{E}[G(x_{k+1}) - G_*] - \frac{\alpha\hat{L}M}{2c\mu} \leq (1 - \mu\hat{\alpha}c)\mathbb{E}[G(x_k) - G_*] + \frac{1}{2}\hat{\alpha}^2 LM - \frac{\alpha\hat{L}M}{2c\mu} =$$

$$= (1 - \mu\hat{\alpha}c)\Big(\mathbb{E}[G(x_k) - G_*] - \frac{\alpha\hat{L}M}{2c\mu}\Big)$$

következik. Majd vigyük át a legutóbb levont konstanst a jobb oldalra és használjuk az előző képlet iterált verzióját:

$$\mathbb{E}[G(x_{k+1}) - G_*] \leq \frac{\alpha\hat{L}M}{2c\mu} + (1 - \mu\hat{\alpha}c)\Big(\mathbb{E}[G(x_k) - G_*] - \frac{\alpha\hat{L}M}{2c\mu}\Big) \leq$$

$$\leq \frac{\alpha\hat{L}M}{2c\mu} + (1 - \mu\hat{\alpha}c)^k\Big(G(x_1) - G_* - \frac{\alpha\hat{L}M}{2c\mu}\Big).$$

Ezzel bizonyítottuk az egyenlőtlenséget. Vegyük észre, hogy a lépéshosszról szóló feltétel alapján:

$$0 < \mu\hat{\alpha}c \leq \frac{\mu^2 c}{LM_G}.$$

Használjuk fel hogy $0 < \mu^2 \leq M_V + \mu_G^2 = M_G$, amiből:

$$0 < \mu\hat{\alpha}c \leq \frac{\mu^2 c}{LM_G} \leq \frac{c}{L} \leq 1.$$

Ezért:

$$\lim_{k \longrightarrow \infty} \frac{\hat{\alpha}LM}{2c\mu} + (1 - \mu\hat{\alpha}c)^k\Big(G(x_1) - G_* - \frac{\hat{\alpha}LM}{2c\mu}\Big) = \frac{\alpha\hat{L}M}{2c\mu}.$$

Ezzel a tételt beláttuk. □

Megfigyelhetjük, hogy minél kissebb értéket választunk $\hat{\alpha}$ értéknek, annál kissebb környezetébe konvergálunk az optimális megoldásnak. Ellenben ekkor $(1 - \mu\hat{\alpha}c)$ együttható nőni fog, ezáltal a konvergencia lassabb lesz. Ezen megfigyelésekből jön az ötlet, hogy nem fix lépéshosszúságot választunk, hanem az iteráció során módosítjuk.
Jelöljük G_α-val, az α lépéshossz választás esetén azt a sugarat, amin belülre érünk az optimális értéktől, azaz $G_\alpha = \frac{\alpha LM}{2c\mu}$. Válasszunk egy α_1 kezdeti lépéhosszt, amire teljesül az 5.1 Tételben előírt feltétel, azaz $0 < \alpha_1 < \frac{\mu}{LM_G}$.
Folytassuk addig a pontig az iterációt ezzel a lépéshosszal, amíg $\mathbb{E}[G(x_{k_2}) - G_*] \leq 2G_{\alpha_1}$. Tehát x_{k_2} jelöli azt az iterációs pontot ahol ez bekövetkezik, ennek megfelelően $x_{k_1} = 1$. Amint elértük ezt a határt $\alpha_2 := \frac{\alpha_1}{2}$-re frissítjük az iterációs lépést. Ezt általánosítva x_{k_r} az az iteráció amelynél először $\mathbb{E}[G(x_{k_r}) - G_*] \leq 2G_{\alpha_{r-1}}$. Illetve $\alpha_r = \alpha_1 2^{1-r}$. Mivel:

$$G_{\alpha_r} = \frac{\alpha_r LM}{2c\mu} = \frac{\alpha_1 2^{1-r} LM}{2c\mu}, \tag{1}$$

ezért:

$$\lim_{r\longrightarrow\infty} G_{\alpha_r} = \lim_{r\longrightarrow\infty} \frac{\alpha_1 2^{1-r} LM}{2c\mu} = 0.$$

Ezzel az eljárással várhatóan az optimumba konvergál az iteráció. Felírhatjuk egy adott nagyságú iterációkhoz tartozó lépésszámokat. Induljunk ki az 5.1-es Tétel állításából, és nézzük meg x_{k_r} és $x_{k_{r+1}}$ közötti összefüggést:

$$\mathbb{E}[G(x_k) - G_*] \leq \frac{\hat{\alpha} LM}{2c\mu} + (1 - \hat{\alpha} c\mu)^{k-1}\Big(G(x_1) - G_* - \frac{\hat{\alpha} LM}{2c\mu}\Big).$$

Ebből x_{k_r}-ből indítva $x_{k_{r+1}}$-ig:

$$\mathbb{E}[G(x_{k_{r+1}}) - G_*] \leq \frac{\alpha_r LM}{2c\mu} + (1 - \alpha_r c\mu)^{k_{r+1}-k_r}\Big(\mathbb{E}[G(x_{k_r}) - G_*] - \frac{\alpha_r LM}{2c\mu}\Big). \tag{2}$$

Használjuk fel, hogy:

$$\mathbb{E}[G(x_{k_r}) - G_*] \leq 2G_{\alpha_{r-1}} = 4G_{\alpha_r},$$
$$\mathbb{E}[G(x_{k_{r+1}}) - G_*] \leq 2G_{\alpha_r}.$$

Ezeket és az (1)-es formulát a (2)-ba helyettesítve kapjuk, hogy:

$$\mathbb{E}[G(x_{k_{r+1}}) - G_*] \leq G_{\alpha_r} + (1 - \alpha_r c\mu)^{k_{r+1}-k_r}(4G_{\alpha_r} - G_{\alpha_r}).$$

Tegyük fel, hogy k_r értéket ismerjük. Azt is tudjuk hogy:

$$\lim_{k_{r+1}\longrightarrow\infty} (1 - \alpha_r c\mu)^{k_{r+1}-k_r} = 0.$$

Ezáltal létezik olyan k_{r+1} index amelyre:

$$G_{a_r} + (1 - \alpha_r c\mu)^{k_{r+1}-k_r}(4G_{a_r} - G_{a_r}) \leq 2G_{\alpha_r}.$$

Becsüljük ennek segítségével a k_{r+1} index értékét. Átrendezve a fenti egyenlőtlenséget:

$$(1 - \alpha_r c\mu)^{k_{r+1}-k_r}(3G_{a_r}) \leq G_{\alpha_r},$$

$$(1 - \alpha_r c\mu)^{k_{r+1}-k_r} \leq \frac{1}{3},$$

$$k_{r+1} - k_r \leq \frac{\log(\frac{1}{3})}{\log(1 - \alpha_r c\mu)} \approx \frac{\log(3)}{\alpha_r c\mu} = O(2^r).$$

Láthatjuk, hogy mindig amikor megfelezzük a lépéshosszúságot, akkor a lépészám megduplázódik. Robbins és Monro [1] eredményei sokkal rugalmasabb feltételeket szabnak a konvergencia biztosításához:

$$\sum_{k=1}^{\infty} \alpha_k = \infty, \qquad \sum_{k=1}^{\infty} \alpha_k^2 < \infty.$$

5.2. Tétel. Legyen $G : \mathbb{R}^d \longrightarrow R$ folytonosan differenciálható és erősen konvex. Továbbá $\nabla G(x)$ Lipschitz-folytonos és a sztochasztikus gradiens iteráció teljesítse az $(A1) \longrightarrow (A3)$ feltételeket. Legyen minden $k \in \mathbb{N}$ esetén:

$$\alpha_k = \frac{\beta}{\gamma + k}, \qquad ahol \qquad \beta > \frac{1}{c\mu}, \qquad \alpha_1 < \frac{\mu}{LM_G}.$$

Ekkor $\forall k \in \mathbb{N}$ esetén, a várható eltérés az optimumtól:

$$\mathbb{E}[G(x_k) - G_*] \leq \frac{\nu}{\gamma + k},$$

ahol:

$$\nu := \max\Big\{\frac{\beta^2 LM}{2(\beta c\mu - 1)}, (\gamma + 1)(G(x_k) - G_*)\Big\}.$$

5.7. Bizonyítás. Használjuk föl, hogy $\alpha_k LM_G \leq \alpha_1 LM_G \leq \mu : \forall k \in N$, továbbá tekintsük az 5.3-as és 5.4-es Lemmákat:

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - G(x_k) \leq -(\mu - \frac{1}{2}\alpha_k LM_G)\alpha_k \big\|\nabla G(x_k)\big\|_2^2 + \frac{1}{2}\alpha_k^2 LM,$$

$$2c(G(x) - G_*) \leq \nabla \big\|G(x)\big\|_2^2 .$$

Mivel $\alpha_k LM_G \leq \mu$ ezért:

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - G(x_k) \leq -(\mu - \frac{1}{2}\alpha_k LM_G)\alpha_k \big\|\nabla G(x_k)\big\|_2^2 + \frac{1}{2}\alpha_k^2 LM \leq$$

$$\leq -\frac{1}{2}\mu\alpha_k \big\|\nabla G(x_k)\big\|_2^2 + \frac{1}{2}\alpha_k^2 LM.$$

Felhasználjuk az 5.4-es lemmát:

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - G(x_k) \leq -\frac{1}{2}\mu\alpha_k \big\|\nabla G(x_k)\big\|_2^2 + \frac{1}{2}\alpha_k^2 LM \leq$$

$$\leq -\alpha_k c\mu \big\|G(x_k) - G_*\big\|_2^2 + \frac{1}{2}\alpha_k^2 LM.$$

Vonjunk le mindkét oldalból G_*-ot és vegyük a teljes várható értéket:

$$\mathbb{E}[G(x_{k+1}) - G_*] \leq (1 - \alpha_k c\mu)\mathbb{E}[G(x_k) - G_*] + \frac{1}{2}\alpha_k^2 LM.$$

Most teljes indukciót alkalmazunk. $k = 1$ esetén ν definíciója miatt igaz hogy:

$$\mathbb{E}[G(x_1) - G_*] \leq \frac{\nu}{\gamma + k} = \frac{\nu}{\gamma + 1} = \alpha_k.$$

Tegyük fel, hogy egy adott k-ig igaz az állításunk és tekintsük a $k + 1$-edik lépést:

$$\mathbb{E}[G(x_{k+1}) - G_*] \leq \Big(1 - \frac{\beta c\mu}{\gamma + k}\Big)\Big(\frac{\nu}{\gamma + k}\Big) + \frac{\beta^2 LM}{2(\gamma + k)^2} =$$

Felhasználva ν definícióját következik, hogy $-\left(\frac{\beta c\mu - 1}{(\gamma+k)^2}\right)\nu + \frac{\beta^2 LM}{2(\gamma+k)^2} \leq 0$ ebből:

$$= \left(\frac{\gamma + k - \beta c\mu}{(\gamma+k)^2}\right)\nu + \frac{\beta^2 LM}{2(\gamma+k)^2} = \left(\frac{\gamma + k - 1}{(\gamma+k)^2}\right)\nu - \left(\frac{\beta c\mu - 1}{(\gamma+k)^2}\right)\nu + \frac{\beta^2 LM}{2(\gamma+k)^2} \leq \frac{\nu}{\gamma + (k+1)}.$$

Így az állítást beláttuk. □

5.4. A sztochasztikus gradiens vizsgálata általános függvények esetén

Az alkalmazott területeken, mint például a neurális hálózatok esetén, sajnos nincs olyan szerencsénk, hogy erősen konvex függvényekről beszéljünk. Ebben a fejezetben megvizsgáljuk a sztochasztikus gradiens módszert általános függvényekre alkalmazva, fix és változó lépéshossz esetén is, ahogy az előző fejezetben láthattuk.

5.3. Tétel. Legyen $G : \mathbb{R}^d \longrightarrow \mathbb{R}$ folytonosan differenciálható függvény, mely teljesíti az $(A1) \longrightarrow (A3)$ feltételeket, továbbá legyen $\nabla G(x)$ mindenütt Lipschitz folytonos. A lépéshossz $\hat{\alpha}$ fix értékű, melyre:

$$0 < \hat{\alpha} < \frac{\mu}{LM_G}$$

teljesül. Ekkor az iteráció pontjaiban teljesül, hogy:

$$\mathbb{E}\Big[\sum_{k=1}^{K}\big\|\nabla G(x_k)\big\|_2^2\Big] \leq \frac{K\hat{\alpha}LM}{\mu} + \frac{2(G(x_1) - G_{inf})}{\mu\hat{\alpha}},$$

ebből következően:

$$\lim_{K\longrightarrow\infty} \mathbb{E}\Big[\frac{1}{K}\sum_{k=1}^{K}\big\|\nabla G(x_k)\big\|_2^2\Big] \leq \lim_{K\longrightarrow\infty} \frac{\hat{\alpha}LM}{\mu} + \frac{2(G(x_1) - G_{inf})}{K\mu\hat{\alpha}} = \frac{\hat{\alpha}LM}{\mu},$$

ahol G_{inf} az $(A1) \longrightarrow (A3)$ feltételeknél meghatározott alsó korlát.

5.8. Bizonyítás. Használjuk fel az 5.3 Lemmát, illetve az $\hat{\alpha}$ értékére kiszabott feltételt:

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - \mathbb{E}[G(x_k)] \leq -(\mu - \frac{1}{2}\hat{\alpha}LM_G)\hat{\alpha}\mathbb{E}[\big\|\nabla G(x_k)\big\|_2^2] + \frac{1}{2}\hat{\alpha}^2LM \leq$$

$$\leq -\frac{1}{2}\mu\hat{\alpha}\mathbb{E}[\big\|\nabla G(x_k)\big\|_2^2] + \frac{1}{2}\hat{\alpha}^2LM.$$

Használjuk, hogy az előzőeket összeadva teleszkópikus összeget nyerünk, amiből adódik, hogy:

$$G_{inf} - G(x_1) \leq \mathbb{E}[G(x_{K+1})] - G(x_1) \leq -\frac{1}{2}\mu\hat{\alpha}\mathbb{E}\Big[\sum_{k=1}^{K}\big\|\nabla G(x_k)\big\|_2^2\Big] + \frac{1}{2}\hat{\alpha}^2KLM,$$

amiből egyszerű átrendezéssel megkapjuk, hogy:

$$\mathbb{E}\Big[\sum_{k=1}^{K}\big\|\nabla G(x_k)\big\|_2^2\Big] \leq \frac{K\hat{\alpha}LM}{\mu} + \frac{2(G(x_1) - G_{inf})}{\mu\hat{\alpha}}.$$

Így a tételt beláttuk. □

5.4. Tétel. Legyen $G : \mathbb{R}^d \longrightarrow \mathbb{R}$ folytonosan differenciálható függvény mely teljesíti az $(A1) \longrightarrow (A3)$ feltételeket, továbbá legyen $\nabla G(x)$ Lipschitz folytonos. A lépéshosszok sorozatát jelölje α_k melyre teljesüljek az alábbi feltételek minden $k \in \mathbb{N}$ esetén:

$$\sum_{k=1}^{\infty} \alpha_k = \infty, \qquad \sum_{k=1}^{\infty} \alpha_k^2 < \infty, \qquad 0 < \alpha_k < \frac{\mu}{LM_G}.$$

Legyen $A_K = \sum_{k=1}^{K} \alpha_k$, ekkor:

$$\lim_{K \longrightarrow \infty} \mathbb{E}\Big[\sum_{k=1}^{K} \alpha_k \big\|\nabla G(x_k)\big\|_2^2\Big] < \infty.$$

Felhasználva hogy $\sum_{k=1}^{\infty} \alpha_k = \infty$ adódik, hogy:

$$\lim_{K \longrightarrow \infty} \mathbb{E}\Big[\frac{1}{A_K}\sum_{k=1}^{K} \alpha_k \big\|\nabla G(x_k)\big\|_2^2\Big] = 0.$$

5.9. Bizonyítás. Használjuk fel az 5.3 Lemmát, továbbá, hogy az α_k értékekre igaz hogy: $0 < \alpha_k < \frac{\mu}{LM_G}$. Az előző bizonyításhoz hasonlóan kapjuk, hogy:

$$\mathbb{E}_{\xi_k}[G(x_{k+1})] - \mathbb{E}[G(x_k)] \leq -(\mu - \frac{1}{2}\alpha_k LM_G)\alpha_k \mathbb{E}[\big\|\nabla G(x_k)\big\|_2^2] + \frac{1}{2}\alpha_k^2 LM \leq$$

$$\leq -\frac{1}{2}\mu\alpha_k \mathbb{E}[\big\|\nabla G(x_k)\big\|_2^2] + \frac{1}{2}\alpha_k^2 LM.$$

Ismét a fentieket összeadva adódik, hogy:

$$G_{inf} - G(x_1) \leq \mathbb{E}[G(x_{K+1})] - G(x_1) \leq -\frac{1}{2}\mu \sum_{k=1}^{K} \alpha_k \mathbb{E}[\big\|\nabla G(x_k)\big\|_2^2] + \frac{1}{2}LM \sum_{k=1}^{K} \alpha_k^2,$$

amiből egyszerű átrendezéssel megkapjuk, hogy:

$$\mathbb{E}\Big[\sum_{k=1}^{K} \alpha_k \big\|\nabla G(x_k)\big\|_2^2\Big] \leq \frac{LM}{\mu}\sum_{k=1}^{K} \alpha_k^2 + \frac{2(G(x_1) - G_{inf})}{\mu}.$$

Használjunk fel, hogy $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$ amiből már láthatjuk, hogy a határérték:

$$\lim_{K \longrightarrow \infty} \mathbb{E}\Big[\sum_{k=1}^{K} \alpha_k \big\|\nabla G(x_k)\big\|_2^2\Big] < \infty.$$

Így a tételt beláttuk. □

6. Szimuláció

6.1. A batch gradiens görbe egyenletes konvergenciájának szimulálása

A 4.1 fejezetben bebizonyítottuk, hogy megfelelő simasági tulajdonságú $\hat{G} : \mathbb{R}^d \longrightarrow \mathbb{R}$ függvényen alkalmazott fix lépéshosszúságú gradiens módszer által adódó folytonos töröttvonal egyenletesen konvergens, határértéke pedig $\gamma^{(l)}(t)$. Szimuláljuk le Matlab segítségével ezt a konvergenciát. Legyen most G függvény $G : \mathbb{R}^2 \longrightarrow \mathbb{R}$ és az általánosítás kedvéért:

$$G(x, y) = \sum_{i=1}^{10} g_i(x, y)$$

alakban írjuk fel, ahol $g_i : \mathbb{R}^2 \longrightarrow \mathbb{R} : \forall i \in \{1, 2, ..., 10\}$. A szimulációban alkalmazott függvény(ek):

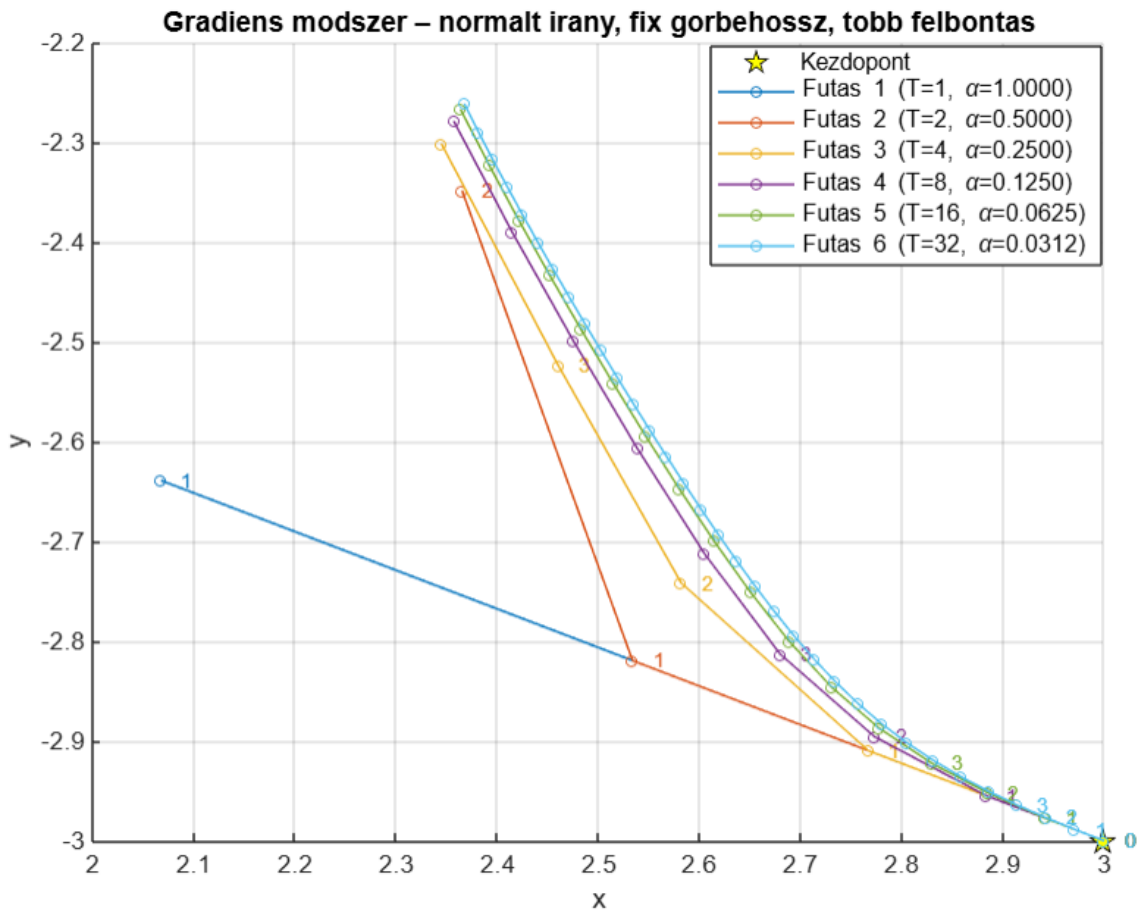
$$g_1(x, y) = \exp(0.5 \sin(2x) + 0.5 \cos(2y)), \qquad g_2(x, y) = \sin(5x) \cos(5y),$$
$$g_3(x, y) = (x^2 + y^2) \sin(x) \cos(y), \qquad g_4(x, y) = \ln(1 + (3x)^2 + (2y)^2),$$
$$g_5(x, y) = (x^4 + y^4) - 2x^2y^2, \qquad g_6(x, y) = 3 \sin(x^2 + y^2) + 0.2(x + y)^2,$$
$$g_7(x, y) = \exp(0.2x^2 + 0.3y^2) \sin(2x + 2y), \qquad g_8(x, y) = (x + 1)^2 + (y - 1)^4 + 0.1 \sin(5x),$$
$$g_9(x, y) = (x - 2)^2 + (y + 3)^2 + 0.5 \sin(3xy), \qquad g_{10}(x, y) = \tanh(xy) + 0.1(x^2 + y^2).$$

A batch gradienst futtassuk le az alábbi paraméterekkel, ahol l a görbe hossza, (x_1, y_1) a kezdőpont, T a különböző lépéshosszok esetén az iterációk száma, α pedig a lépéshossz:

$$l = 1,$$
$$(x_1, y_1) = (3, -3),$$
$$T \in \{1, 2, 4, 8, 16, 32\},$$
$$\alpha \in \{1, 0.5, 0.25, 0.125, 0.0625, 0.03125\}.$$

A belátott tétel alapján, azt várjuk, hogy a lépéshossz csökkentésével a görbénk egyre inkább rásimul, egy sima görbére. A lépéshosszt minden lefuttatott batch gradiens után felezni fogjuk, tehát exponenciálisan csökkentjük, ebből adódóan arra számíthatunk, hogy viszonylag gyorsan fog konvergálni ehhez a görbéhez. A G függvény komponenseit, úgy választottam meg, hogy érzékeny legyen a lépéshosszra, tartalmaz magasabb fokú polinomokat, illetve trigonometrikus tagokat.

A Matlab szimuláció az alábbi eredménnyel zárult (6. ábra):



6. ábra. A fix lépéshosszúságú batch gradiens görbe konvergenciájának szimulációja.

A szimuláció azt az eredményt adta, amire számítottunk. Az első két batch gradiens, csupán 1 és 2 lépésből állt, láthatjuk, hogy ez mennyire eltér a közelített görbétől. A 4. 5. és 6. kisebb lépéshosszúságú futtatás pedig jól láthatóan simul rá a közelített görbére.

A szimulációt megvalósító Matlab kódot megtekithetjük a 7. 8. és 9. ábrákon:

```
function gradient_descent_fixed_curve(g_funcs, x0, L, k, num_runs)
    % g_funcs: 10 darab g_i(x, y) fuggveny (cell array)
    % x0: kezdopont [x; y]
    % L: teljes gorbehossz, amit egy futas alatt bejarunk
    % k: osztasi tenyezo (pl. 2)
    % num_runs: hany iteraciot fusson kulonbozo felbontassal

    n = length(g_funcs); % fuggvenyek szama
```

7. ábra. Fix lépéshosszúságú batch gradiens konvergencia szimulációs kód. (1.részlet)

```
figure;
clf;
hold on;
colors = lines(num_runs); % kulonbozo szinek minden futashoz

% A kezdopont kulon is megjelenik
plot(x0(1), x0(2), 'kp', 'MarkerSize', 10, 'MarkerFaceColor', 'y', ...
    'DisplayName', 'Kezdopont');

for run = 0:(num_runs - 1)
    T = k^run;            % lepesek szama
    alpha = L / T;        % lepeshossz

    x = x0;               % kezdoopont minden futasnal
    history = zeros(2, T + 1);
    history(:, 1) = x0;   % kezdoopont is benne van az utvonalban

    for iter = 1:T
        grad_total = zeros(2, 1);

        % Gradiens kiszamitasa az osszes g_i(x,y) fuggvenyre
        for i = 1:n
            grad_total = grad_total + numerical_gradient(g_funcs{i}, x);
        end

        grad_avg = grad_total / n;

        % Iranygradiens normalizalasa (egyseg hosszusagu)
        grad_dir = grad_avg / (norm(grad_avg) + 1e-12);

        % Gradiens lepes
        x = x - alpha * grad_dir;

        % Taroljuk a lepest
        history(:, iter + 1) = x;
    end

    % Utvonal megjelenitese
    plot(history(1, :), history(2, :), '-o', ...
        'Color', colors(run + 1, :), ...
        'MarkerSize', 4, ...
        'DisplayName', sprintf('Futas %d (T=%d, \\alpha=%.4f)', run + 1, T,
            alpha));

    % Pontok megjelolese (kezdoopont is benne van most mar)
    for j = 1:min(4, size(history, 2))
        text(history(1, j) + 0.02, history(2, j), sprintf('%d', j - 1), ...
            'Color', colors(run + 1, :), 'FontSize', 8);
```

8. ábra. Fix lépéshosszúságú batch gradiens konvergencia szimulációs kód. (2.részlet)

```
        end
    end

    % Abra beallitasok
    xlabel('x'); ylabel('y');
    title('Gradiens modszer, normalt irany, fix gorbehossz, tobb felbontas');
    legend show;
    grid on;
end


function grad = numerical_gradient(f, x)
    % Ketvaltozos fuggveny numerikus gradiense
    h = 1e-6;
    fx = f(x(1), x(2));
    dfdx = (f(x(1) + h, x(2)) - fx) / h;
    dfdy = (f(x(1), x(2) + h) - fx) / h;
    grad = [dfdx; dfdy];
end

% 10 db celfuggveny
g_funcs = {
    @(x, y) exp(0.5 * sin(2*x) + 0.5 * cos(2*y));          % hullamzo, sima
        exponencialis
    @(x, y) sin(5*x) .* cos(5*y);                          % gyorsan valto
        gradiens, sima
    @(x, y) (x.^2 + y.^2) .* sin(x) .* cos(y);             % nemlinearis
        szorzat, C2
    @(x, y) log(1 + (3*x).^2 + (2*y).^2);                  % logaritmikus, de
        sima es Lipschitz gradfuggvenyu
    @(x, y) (x.^4 + y.^4) - 2*(x.^2).*(y.^2);              % saddle-jellegu, C2
    @(x, y) 3*sin(x.^2 + y.^2) + 0.2*(x + y).^2;           % gyors
        gradiensvaltas, sima
    @(x, y) exp(0.2*x.^2 + 0.3*y.^2) .* sin(2*x + 2*y);    % erzekeny,
        oszcillalo exponencialissal
    @(x, y) (x + 1).^2 + (y - 1).^4 + 0.1*sin(5*x);        % aszimmetrikus +
        hullamzas
    @(x, y) (x - 2).^2 + (y + 3).^2 + 0.5*sin(3*x.*y);     % parabola, trigonos
        modulacioval
    @(x, y) tanh(x.*y) + 0.1*(x.^2 + y.^2);                % sima, nemlinearis,
        laposodo szelek
};

x0 = [3; -3];   % Kezdopont
L = 1.0;        % Fix gorbehossz minden iteraciora
k = 2;          % Lepesszam osztas
num_runs = 6;   % Iteraciok kulonbozo T ertekkel

gradient_descent_fixed_curve(g_funcs, x0, L, k, num_runs);
```

9. ábra. Fix lépéshosszúságú batch gradiens konvergencia szimulációs kód. (3.részlet)

6.2. A batch és a sztochasztikus gradiens konvergenciájának szimulálása

Ebben a fejezetben szimulálni fogjuk, az optimumba való konvergenciát változó lépéshosszúság esetén, mind a batch, mind a sztochasztikus gradiens esetén. Az előző fejezethez hasonlóan legyen ismét $G : \mathbb{R}^2 \longrightarrow \mathbb{R}$ függvényünk az alábbi alakú:

$$G(x, y) = \sum_{i=1}^{10} g_i(x, y)$$

ahol g_i függvények az alábbiak:

$$\begin{aligned}
g_1(x, y) &= (x-1)^2 + (y-2)^2, & g_2(x, y) &= (x+1)^2 + (y+1)^2, \\
g_3(x, y) &= \sin(x) + \cos(y), & g_4(x, y) &= \exp(0.1x^2 + 0.1y^2), \\
g_5(x, y) &= x^2 + y^2, & g_6(x, y) &= (x-2)^2 + (y+2)^2, \\
g_7(x, y) &= \ln(1 + x^2 + y^2), & g_8(x, y) &= (x+3)^2 + (y-1)^2, \\
g_9(x, y) &= xy + y^2, & g_{10}(x, y) &= |x| + |y|.
\end{aligned}$$

A változó lépéshosszúságot több módszerrel is számolhatjuk, néhány ezen módszerek közül:

(M1): Inverse square root: $\alpha_k = \alpha_{k-1}/\sqrt{k}$

(M2): Inverse time decay: $\alpha_k = \alpha_{k-1}/(1 + \hat{\beta}k)$

(M3): Exponential : $\alpha_k = \alpha_{k-1}\hat{\gamma}^k$ ahol $\hat{\gamma} \in (0, 1)$

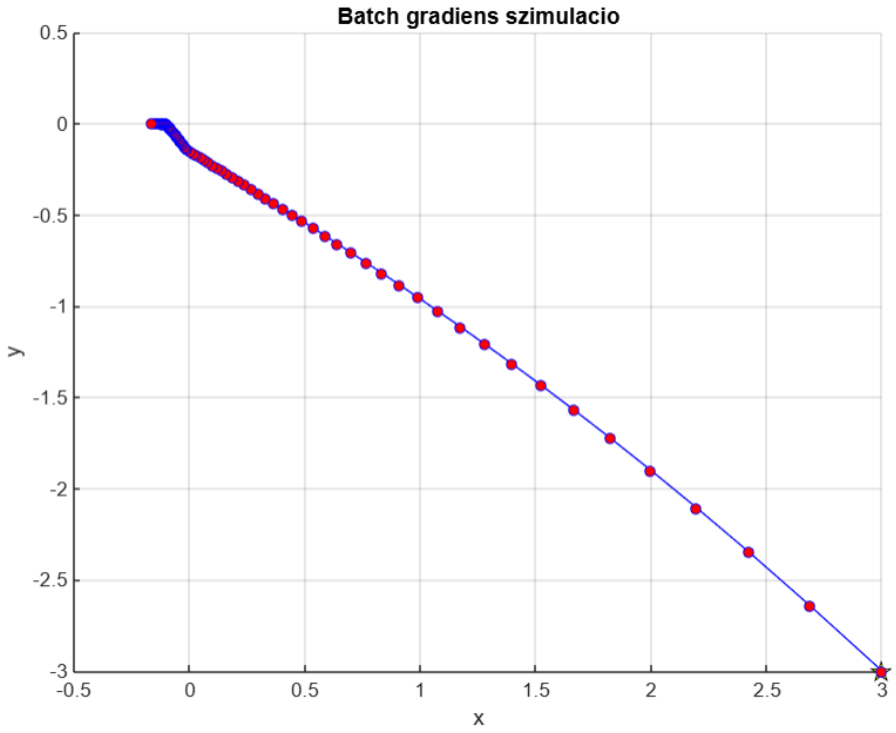
A szimulációt az inverse time decay módszerrel fogjuk végezni, továbbá az alábbi paramétereket használjuk:

$$\begin{aligned}
n &= 100, & (x_1, y_1) &= (3, -3), & \hat{\beta} &= 0.05, \\
\alpha_1 &= 0.1,
\end{aligned}$$

ahol n az iterációk száma.

A sztochasztikus gradiens szimulációja esetén, minden iterációban véletlenszerűen választunk két függvényt a tíz közül és aszerint számoljuk majd ki a gradiens értékét. Ez azt jelenti, hogy minden egyes iterációban 10 gradiens kiszámolása helyett, kettőt számolunk ki, azaz a számítási költsége egy lépésnek ötöde a batch gradiens módszeréhez képest. Nagy neurális hálózatok esetén, ez hatalmas előnyt jelent, hiszem a backpropagation algoritmus végrehajtási száma töredéke a batch gradienséhez viszonyítva.

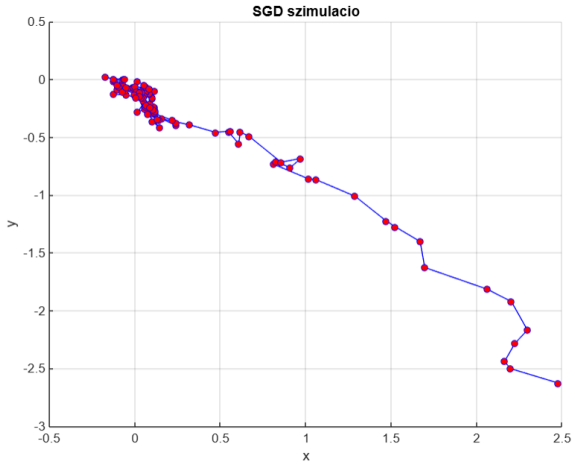
Tekintsük először a batch gradiens szimuláció eredményét (10. ábra):



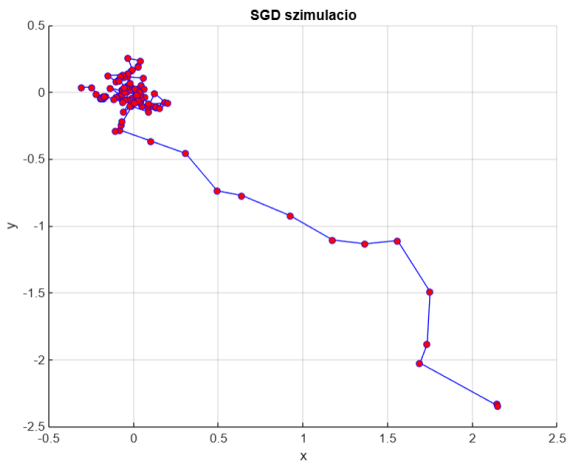
10. ábra. A batch gradiens konvergenciájának szimulációja.

Megfigyelhetjük, hogy az batch gradiens közelít egy lokális minimumhelyet a $(0, 0)$ pont, egy kicsi környezetében. Mivel a gradiens egy lokális minimumhoz közelítve megfelelő simaságú függvény mellett tart a nullába, továbbá az α_k szintén tart a nullába, ezért a konkrét lépéshossz, amit lép az iteráció, lépésről lépésre csökken.

Tekintsük most a sztochasztikus gradiens iterációját (11. és 12. ábra):

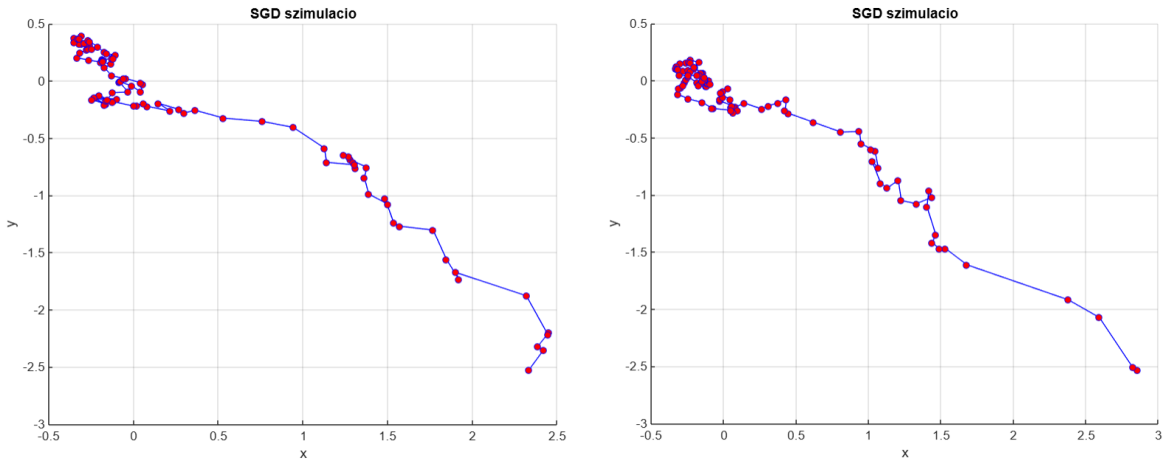


(a) Első szimuláció



(b) Második szimuláció

11. ábra. A sztochasztikus gradiens konvergenciájának szimulációja. (1-2)



(a) Harmadik szimuláció (b) Negyedik szimuláció

12. ábra. A sztochasztikus gradiens konvergenciájának szimulációja. (3-4)

Ellentétben a batch gradiens módszerrel, itt a konkrét lépéshosszok nem csökkennek lépésről lépésre, hanem előfordulhat növekedés is. Ennek az az oka, hogy nem vesszük figyelembe az összes függvény szerinti gradiens értékét, ezért nagyobb mértékben térhetünk el a valódi optimális iránytól. Ezáltal a gradiens hossza könnyedén növekedhet is, ami ellenében hathat a csökkenő α_k értékeknek.

A batch gradiens adatai az 50., 75. és 100. iterációban:

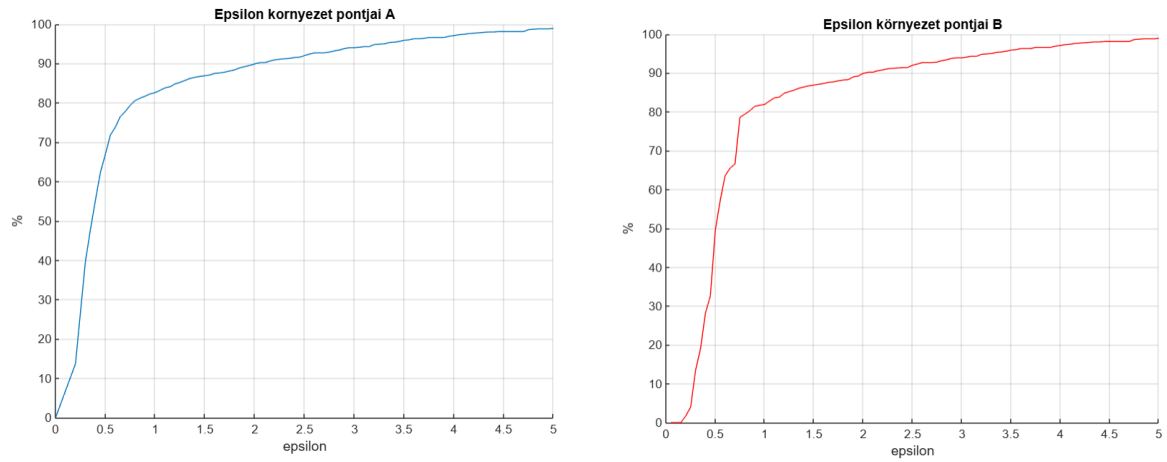
- 50. iteráció: x = -0.0554, y = -0.0680, lépéshossz = 0.0286
- 75. iteráció: x = -0.1118, y = -0.0001, lépéshossz = 0.0211
- 100. iteráció: x = -0.1344, y = -0.0000, lépéshossz = 0.0167

A sztochasztikus gradiens adatai az 50., 75. és 100. iterációban:

- 50. iteráció: x = -0.1665, y = -0.0810, lépéshossz = 0.0286
- 75. iteráció: x = -0.3905, y = 0.0345, lépéshossz = 0.0211
- 100. iteráció: x = -0.3946, y = 0.0742, lépéshossz = 0.0167

Az adatokból és a grafikonokból azt a következtetést vonhatjuk le, hogy a sztochasztikus gradiens már az 50. iterációban az optimum közelében volt, majd elkezdett oszcillálni az ennek egy kis környezetében.

Vizsgáljuk meg kétféleképpen, hogy az optimum különböző ϵ környezeteibe, a sztochasztikus gradiens iterációs pontainak hány százaléka esik. Futtassunk 10 sztochasztikus gradiens szimulációt, és nézzük meg a százalékokat összességében (A verzió), másrészt pedig úgy, hogy kikötjük, hogy csak azokat az iterációs pontokat vesszük figyelembe, amelyek után már minden iterációs pont benne van a környezetben (B verzió). Tekintsük a szimulációk eredményét:



(a) "A" szimuláció (b) "B" szimuláció

13. ábra. Az epszilon környezet tartalmazás szimulációja.

A diagramokról leolvashatjuk, hogy a sztochasztikus gradiens hirtelen kezd el konvergálni az optimum felé, majd annak egy környezetében oszcillál. Ez azt jelenti, hogy jelen esetben ötöd annyi gradienst kellett kiszámolni ahhoz, hogy az optimális megoldást viszonylag jól megközelítsük. A gyarkolati alkalmazásokban, mint például a neurális hálózatok esetén, elegendő viszonylag közel kerülni az optimumhoz, ebből láthatjuk, hogy miért is ilyen hatékony és egyben a leggyakrabban használt eszköz a sztochasztikus gradiens módszer a neurális hálózatok tanításánál.

A batch gradiens szimuláció Matlab kódja (14., 15., 16. ábrák):

```
function gradient_descent(g_funcs, x0, num_iters, alpha_0, step_type,
    step_params)
    % g_funcs: cell array, 10 darab g_i(x, y) fuggveny
    % x0: kezdopont [x; y]
    % num_iters: iteraciok szama
    % alpha_0: kezdeti (vagy fix) lepeshossz
    % step_type: 'fixed', 'inv_sqrt', 'inv_linear', 'exp', 'step_decay'
    % step_params: lepeshossz tipusahoz tartozo parameterek

    x = x0; % kezdopont
    n = length(g_funcs); % fuggvenyek szama
    history = zeros(2, num_iters+1); % pontok tarolasa

    history(:, 1) = x0; % kezdopont is benne van az utvonalban

    figure;
    clf;
    hold on;
    % A kezdopont kulon is megjelenik
    plot(x0(1), x0(2), 'kp', 'MarkerSize', 10, 'MarkerFaceColor', 'y', ...
        'DisplayName', 'Kezdopont');

    for iter = 1:num_iters
        grad_total = zeros(2, 1);

        % Osszes g_i gradiense
        for i = 1:n
            grad_total = grad_total + numerical_gradient(g_funcs{i}, x);
        end

        % Atlag gradiens (nem kotelezo, de stabilabb)
        grad_avg = grad_total / n;

        % Lepeshossz szamitas
        step_size = compute_step_size(step_type, alpha_0, iter, step_params);

        % Frissites
        x = x - step_size * grad_avg;

        % Eredmeny mentese
        history(:, iter+1) = x;

        fprintf('Iteracio %d: x = %.4f, y = %.4f, lepeshossz = %.4f\n', iter,
            x(1), x(2), step_size);
    end
```

14. ábra. Batch gradiens szimulációs kód. (1. részlet)

```
% Abra kirajzolasa
    plot(history(1, :), history(2, :), '-o', 'Color', 'b', 'MarkerSize', 5,
        'MarkerFaceColor', 'r');
    xlabel('x'); ylabel('y');
    title('Batch gradiens iteracio');
    grid on;
end

function grad = numerical_gradient(f, x)
    % Numerikus gradiens ketvaltozos fuggvenyhez
    h = 1e-6;
    fx = f(x(1), x(2));
    dfdx = (f(x(1) + h, x(2)) - fx) / h;
    dfdy = (f(x(1), x(2) + h) - fx) / h;
    grad = [dfdx; dfdy];
end

function alpha = compute_step_size(type, alpha_0, t, params)
    % Lepeshossz szamitasa kulonbozo modszerek szerint
    switch lower(type)
        case 'fixed'
            alpha = alpha_0;
        case 'inv_sqrt'
            alpha = alpha_0 / sqrt(t);
        case 'inv_linear'
            beta = getfield_or_default(params, 'beta', 0.1);
            alpha = alpha_0 / (1 + beta * t);
        case 'exp'
            gamma = getfield_or_default(params, 'gamma', 0.99);
            alpha = alpha_0 * gamma^t;
        case 'step_decay'
            gamma = getfield_or_default(params, 'gamma', 0.5);
            k = getfield_or_default(params, 'k', 50);
            alpha = alpha_0 * gamma^floor(t / k);
        otherwise
            error('Ismeretlen lepeshossz tipusa: %s', type);
    end
end

function val = getfield_or_default(structure, field, default)
    % Ha egy mezo nem szerepel a strukturaban, visszaad alapertelmezett erteket
    if isfield(structure, field)
        val = structure.(field);
    else
        val = default;
    end
end
```

15. ábra. Batch gradiens szimulációs kód. (2.részlet)

```
% 10 darab g_i fuggveny definialasa
g_funcs = {
    @(x, y) (x - 1)^2 + (y - 2)^2;
    @(x, y) (x + 1)^2 + (y + 1)^2;
    @(x, y) sin(x) + cos(y);
    @(x, y) exp(0.1*x^2 + 0.1*y^2);
    @(x, y) x^2 + y^2;
    @(x, y) (x - 2)^2 + (y + 2)^2;
    @(x, y) log(1 + x^2 + y^2);
    @(x, y) (x + 3)^2 + (y - 1)^2;
    @(x, y) x * y + y^2;
    @(x, y) abs(x) + abs(y);
};

% Parameterek
x0 = [3; -3];                    % kezdopont
num_iters = 100;                 % iteraciok szama
alpha_0 = 0.1;                   % kezdeti lepeshossz
step_type = 'inv_linear';        % lepeshossz tipusa
step_params = struct('beta', 0.05); % parameter a lepeshosszhoz

% Futtatas
gradient_descent(g_funcs, x0, num_iters, alpha_0, step_type, step_params);
```

16. ábra. Batch gradiens szimulációs kód. (3.részlet)

A sztochasztikus gradiens szimuláció Matlab kódja (17., 18., 19., 20. ábrák):

```
function stochastic_gradient_descent(g_funcs, x0, num_iters, alpha_0,
    step_type, step_params)
    % g_funcs: cell array, 10 darab g_i(x, y) fuggveny
    % x0: kezdopont [x; y]
    % num_iters: iteraciok szama
    % alpha_0: kezdeti (vagy fix) lepeshossz
    % step_type: 'fixed', 'inv_sqrt', 'inv_linear', 'exp', 'step_decay'
    % step_params: lepeshossz tipushoz tartozo parameterek (pl. gamma, beta)

    x = x0; % aktualis pont inicializalasa
    n = length(g_funcs); % g_i fuggvenyek szama
    history = zeros(2, num_iters); % iteracios eredmenyek tarolasa
```

17. ábra. Sztochasztikus gradiens szimulációs kód. (1.részlet)

```
for iter = 1:num_iters
       % Veletlenszeruen kivalasztunk ket kulonbozo fuggvenyt
       idx = randperm(n, 2);

       % Numerikus gradiens szamitasa a kivalasztott fuggvenyekre
       grad1 = numerical_gradient(g_funcs{idx(1)}, x);
       grad2 = numerical_gradient(g_funcs{idx(2)}, x);

       % Atlagolt gradiens
       grad_avg = (grad1 + grad2) / 2;

       % Lepeshossz meghatarozasa az aktualis iteraciohoz
       step_size = compute_step_size(step_type, alpha_0, iter, step_params);

       % Frissites: uj pont szamitasa
       x = x - step_size * grad_avg;

       % Elmentjuk az aktualis pontot
       history(:, iter) = x;

       % Iteracios kiiras
       fprintf('Iteracio %d: x = %.4f, y = %.4f, lepeshossz = %.4f\n', iter,
           x(1), x(2), step_size);
   end

   % Eredmeny kirajzolasa

   figure;
   clf;
   hold on;
   %plot(history(1, :), history(2, :), '-o');
   plot(history(1, :), history(2, :), '-o', 'Color', 'b', 'MarkerSize', 5,
       'MarkerFaceColor', 'r');
   xlabel('x'); ylabel('y');
   title('SGD iteracio');
   grid on;

end

function grad = numerical_gradient(f, x)
   % Ketvaltozos fuggveny numerikus gradiense (kozep-kulonbseggel)
   h = 1e-6;
   fx = f(x(1), x(2));
   dfdx = (f(x(1) + h, x(2)) - fx) / h;
   dfdy = (f(x(1), x(2) + h) - fx) / h;
   grad = [dfdx; dfdy];
end
```

18. ábra. Sztochasztikus gradiens szimulációs kód. (2.részlet)

```
function alpha = compute_step_size(type, alpha_0, t, params)
    % Lepeshossz szamitasa a kivalasztott modszer alapjan
    switch lower(type)
        case 'fixed'
            alpha = alpha_0;

        case 'inv_sqrt'
            alpha = alpha_0 / sqrt(t);

        case 'inv_linear'
            beta = getfield_or_default(params, 'beta', 0.1);
            alpha = alpha_0 / (1 + beta * t);

        case 'exp'
            gamma = getfield_or_default(params, 'gamma', 0.99);
            alpha = alpha_0 * gamma^t;

        case 'step_decay'
            gamma = getfield_or_default(params, 'gamma', 0.5);
            k = getfield_or_default(params, 'k', 50);
            alpha = alpha_0 * gamma^floor(t / k);

        otherwise
            error('Ismeretlen lepeshossz tipusa: %s', type);
    end
end

function val = getfield_or_default(structure, field, default)
    % Parameter alapertelmezett ertekkel, ha hianyzik a strukturabol
    if isfield(structure, field)
        val = structure.(field);
    else
        val = default;
    end
end

% 10 darab celfuggveny definialasa
g_funcs = {
    @(x, y) (x - 1)^2 + (y - 2)^2;
    @(x, y) (x + 1)^2 + (y + 1)^2;
    @(x, y) sin(x) + cos(y);
    @(x, y) exp(0.1*x^2 + 0.1*y^2);
    @(x, y) x^2 + y^2;
    @(x, y) (x - 2)^2 + (y + 2)^2;
    @(x, y) log(1 + x^2 + y^2);
    @(x, y) (x + 3)^2 + (y - 1)^2;
    @(x, y) x * y + y^2;
    @(x, y) abs(x) + abs(y);
};
```

19. ábra. Sztochasztikus gradiens szimulációs kód. (3.részlet)

```
% Parameterek beallitasa
x0 = [3; -3];                  % Kezdeti pont
num_iters = 100;               % Iteraciok szama
alpha_0 = 0.1;                 % Kezdeti lepeshossz
step_type = 'inv_linear';    % Lepeshossz tipusa: fixed, inv_sqrt, inv_linear,
    exp, step_decay
step_params = struct('beta', 0.05); % Csak ha szukseges

% Futtatas
stochastic_gradient_descent(g_funcs, x0, num_iters, alpha_0, step_type,
    step_params);
```

20. ábra. Sztochasztikus gradiens szimulációs kód. (4.részlet)

Az epszilon környezet tartalmazás szimulációs kódja (21., 22., 23. ábrák):

```
function stoch_grad_test(g_funcs, x0, num_iters, alpha_0, step_type,
    step_params, number_of_testruns, optimum, epsilon, eps_res)

%Inicializaljuk az eredmenyt tartalmazo matrixot
result_points(2,num_iters,number_of_testruns) = zeros();

for testrun = 1:number_of_testruns

    %Beirjuk ez eredmenyek koze az aktualis SGD pontokat
    result_points(:,:,testrun)=stochastic_gradient_descent(g_funcs, x0,
        num_iters, alpha_0, step_type, step_params);
end

%Megnzezzuk, hogy az optimum kulonbozo kornyezeteiben hany iteracios pont
%van

%Letrehozzuk, a kulonbozo epszilonokhoz tartozo tombot
percentage_in_epsilon = zeros(2, eps_res);

%Ezzel a mertekkel pasztazzuk vegig az epszilon kornyezetet
eps_step = epsilon/eps_res;

%Aktualis epszilon ertek
eps_curr = epsilon;

%Osszes kiszamolt adat szama
num_of_datas = num_iters*number_of_testruns;
```

21. ábra. Az epszilon környezet tartalmazás szimulációs kód. (1. részlet)

```
%Megszamoljuk hany pont van a kulonbozo kornyezetekben
for eps_count=1:eps_res
    for i=1:number_of_testruns
        for j=1:num_iters
            if norm(optimum-result_points(:,j,i))<eps_curr
                curr_data = percentage_in_epsilon(:,eps_count);
                curr_data = [eps_curr, curr_data(2)+1];
                percentage_in_epsilon(:,eps_count)=curr_data;
            end
        end
    end
    eps_curr = eps_curr-eps_step;
end

%Kiszamoljuk a szazalek erteket

for eps_count = 1:eps_res
    curr_data = percentage_in_epsilon(:,eps_count);
    curr_data = [curr_data(1), curr_data(2)/num_of_datas*100];
    percentage_in_epsilon(:,eps_count) = curr_data;
end

%Letrehozzuk, a kulonbozo epszilonokhoz tartozo tombot
percentage_in_epsilon2 = zeros(2, eps_res);

%Ezzel a mertekkel pasztazzuk vegig az epszilon kornyezetet
eps_step = epsilon/eps_res;

%Aktualis epszilon ertek
eps_curr = epsilon;

%Megszamoljuk hany pont van a kulonbozo kornyezetekben
for eps_count=1:eps_res

    percentage_in_epsilon2(:,eps_count)=[eps_curr,0];

    for i=1:number_of_testruns

        for j=num_iters:-1:1

            if norm(optimum-result_points(:,j,i))<eps_curr
                curr_data = percentage_in_epsilon2(:,eps_count);
                curr_data = [eps_curr, curr_data(2)+1];
                percentage_in_epsilon2(:,eps_count)=curr_data;
            else
                break;
            end

        end
    end
```

22. ábra. Az epszilon környezet tartalmazás szimulációs kód. (2. részlet)

```
eps_curr = eps_curr-eps_step;

end

%Kiszamoljuk a szazalek erteket

for eps_count = 1:eps_res
    curr_data = percentage_in_epsilon2(:,eps_count);
    curr_data = [curr_data(1), curr_data(2)/num_of_datas*100];
    percentage_in_epsilon2(:,eps_count) = curr_data;
end


%Kirajzoljuk az eredmenyt
figure;
clf;
hold on;
plot(percentage_in_epsilon(1, :), percentage_in_epsilon(2, :));
xlabel('epsilon'); ylabel('%');
title('Epsilon kornyezet pontjai A');
grid on;

figure;
clf;
hold on;
plot(percentage_in_epsilon2(1, :), percentage_in_epsilon2(2, :), 'red');
xlabel('epsilon'); ylabel('%');
title('Epsilon Ã¶krnyezet pontjai B');
grid on;
end

% Parameterek beallitasa
x0 = [3; -3];                 % Kezdeti pont
num_iters = 100;              % Iteraciok szama
alpha_0 = 0.1;                % Kezdeti lepeshossz
step_type = 'inv_linear';    % Lepeshossz tipusa: fixed, inv_sqrt, inv_linear,
    exp, step_decay
step_params = struct('beta', 0.05); % Csak ha szukseges
optimum = [-0.1634,0.0001];
number_of_testruns = 10;
epsilon = 5;
eps_res = 100;

stoch_grad_test(g_funcs, x0, num_iters, alpha_0, step_type, step_params,
    number_of_testruns, optimum, epsilon, eps_res);
```

23. ábra. Az epszilon környezet tartalmazás szimulációs kód. (3. részlet)

7. Források

Hivatkozások

[1] Léon Bottou, Frank E. Curtis, Jorge Nocedal (2018) *Optimization Methods for Large-Scale Machine Learning*

[2] Numerical Optimization Line search descent methods
https://indrag49.github.io/Numerical-Optimization/line-search-descent-methods.html

[3] Wikipédia - Training, validation, and test data sets
https://en.wikipedia.org/wiki/Training,_validation,_and_test_data_sets

[4] MIT OpenCourseWare 25. Stochastic Gradient Descent
https://www.youtube.com/watch?v=k3AiUhwHQ28

[5] First Principles of Computer Vision Backpropagation Algorithm | Neural Networks
https://www.youtube.com/watch?v=k3AiUhwHQ28

Alulírott, Mészáros Joshua matematika alapszakos BSc hallgató (ELTE TTK), ezúton nyilatkozom, hogy a szakdolgozatot a témavezetőm irányítása mellett önállóan készítettem, és a dolgozatban egyértelműen feltüntettem minden felhasznált forrást. Minden olyan részt, amely szó szerint vagy azonos, illetve hasonló tartalommal átvett más munkákból származik, hivatkozással jelöltem.

Budapest, 2025. június 1.

Mészáros Joshua

Alulírott, Mészáros Joshua, kijelentem, hogy a AI alapú eszközök közül kizárólag az alábbiakat használtam a dolgozat megírásához:

Feladat	**AI eszköz**	**Felhasználás helye**
Batch gradiens görbe szimulációs Matlab kód generálása	GPT-4o	6.1 fejezet
Batch gradiens szimulációs Matlab kód generálása	GPT-4o	6.2 fejezet
Sztochasztikus gradiens szimulációs Matlab kód generálása	GPT-4o	6.2 fejezet